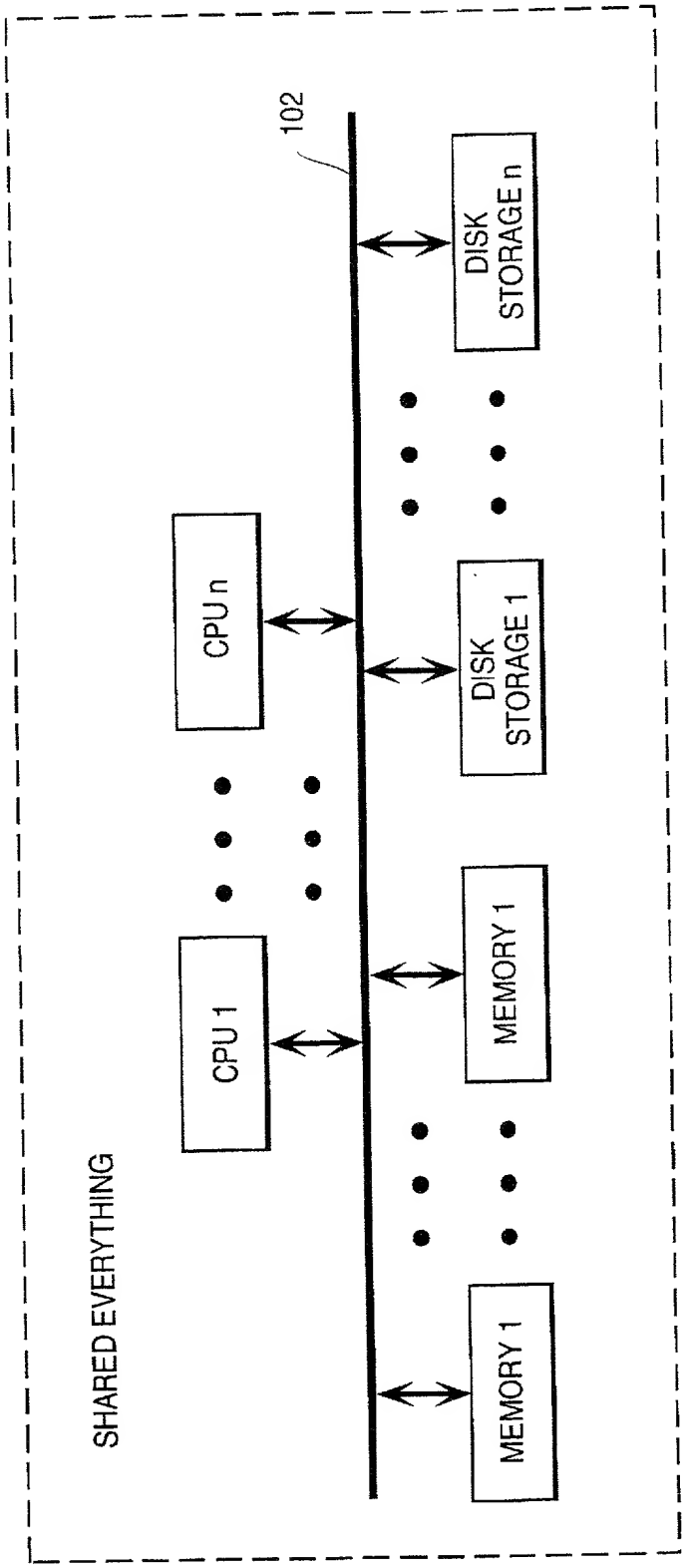

FIG. 1A (PRIOR ART)

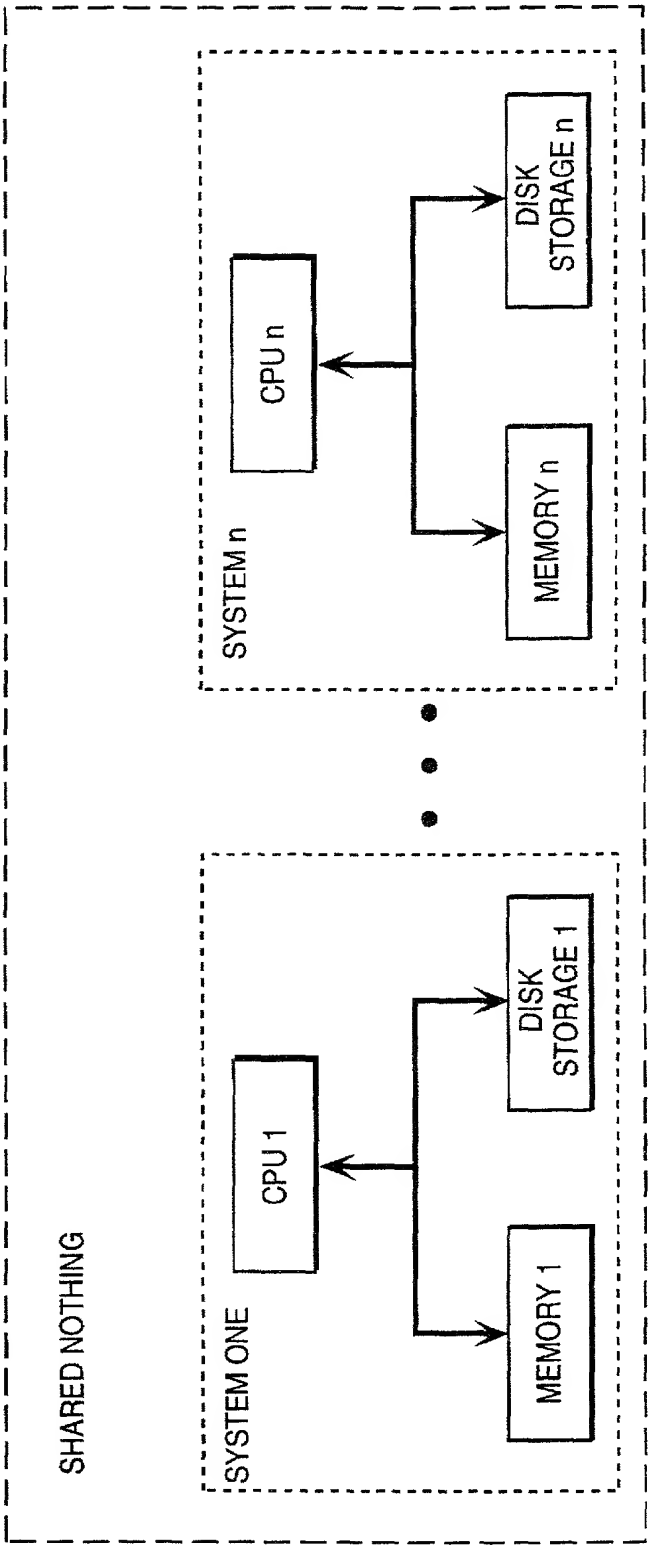

FIG. 1B (PRIOR ART)

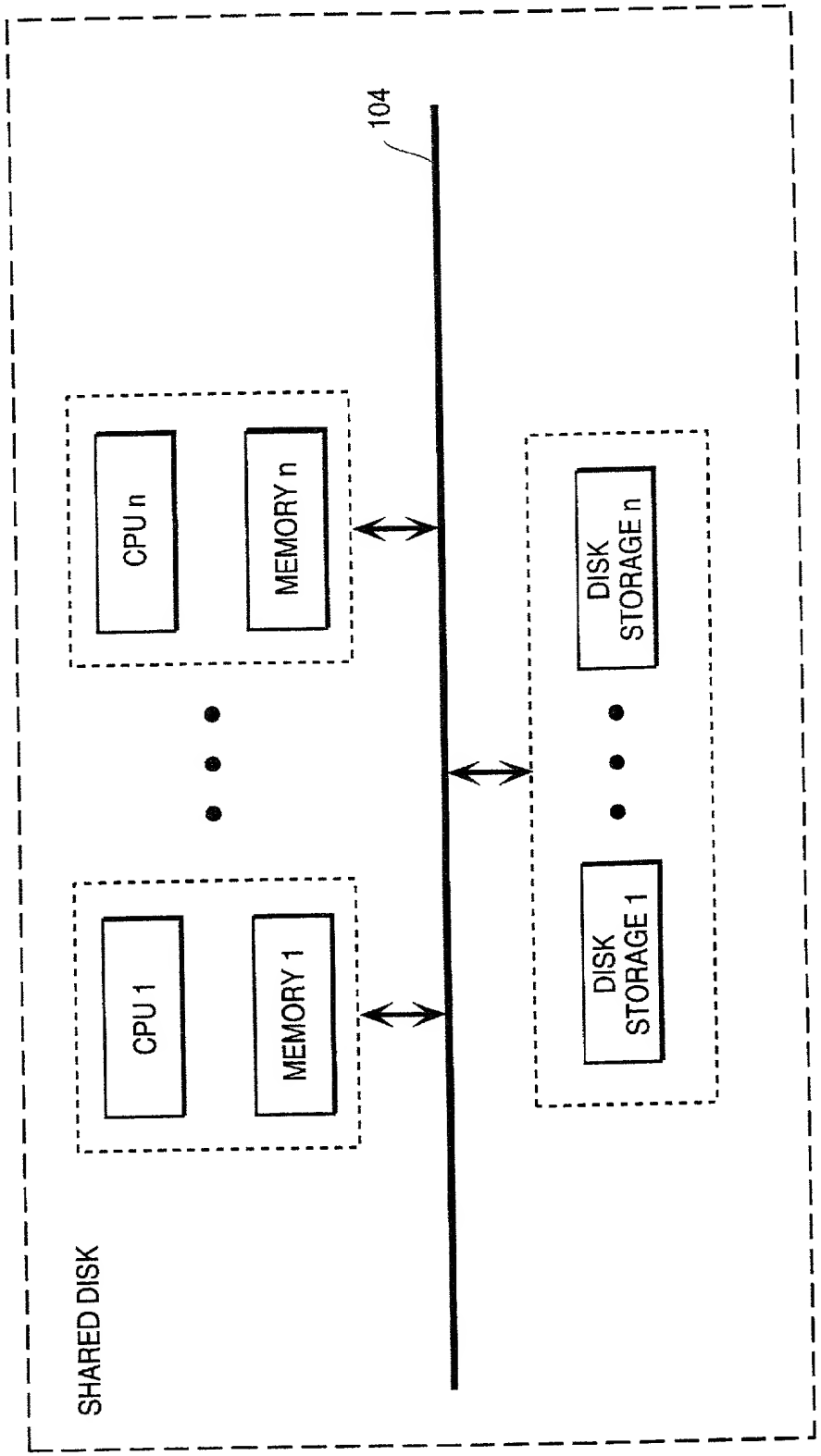

FIG. 1C (PRIOR ART)

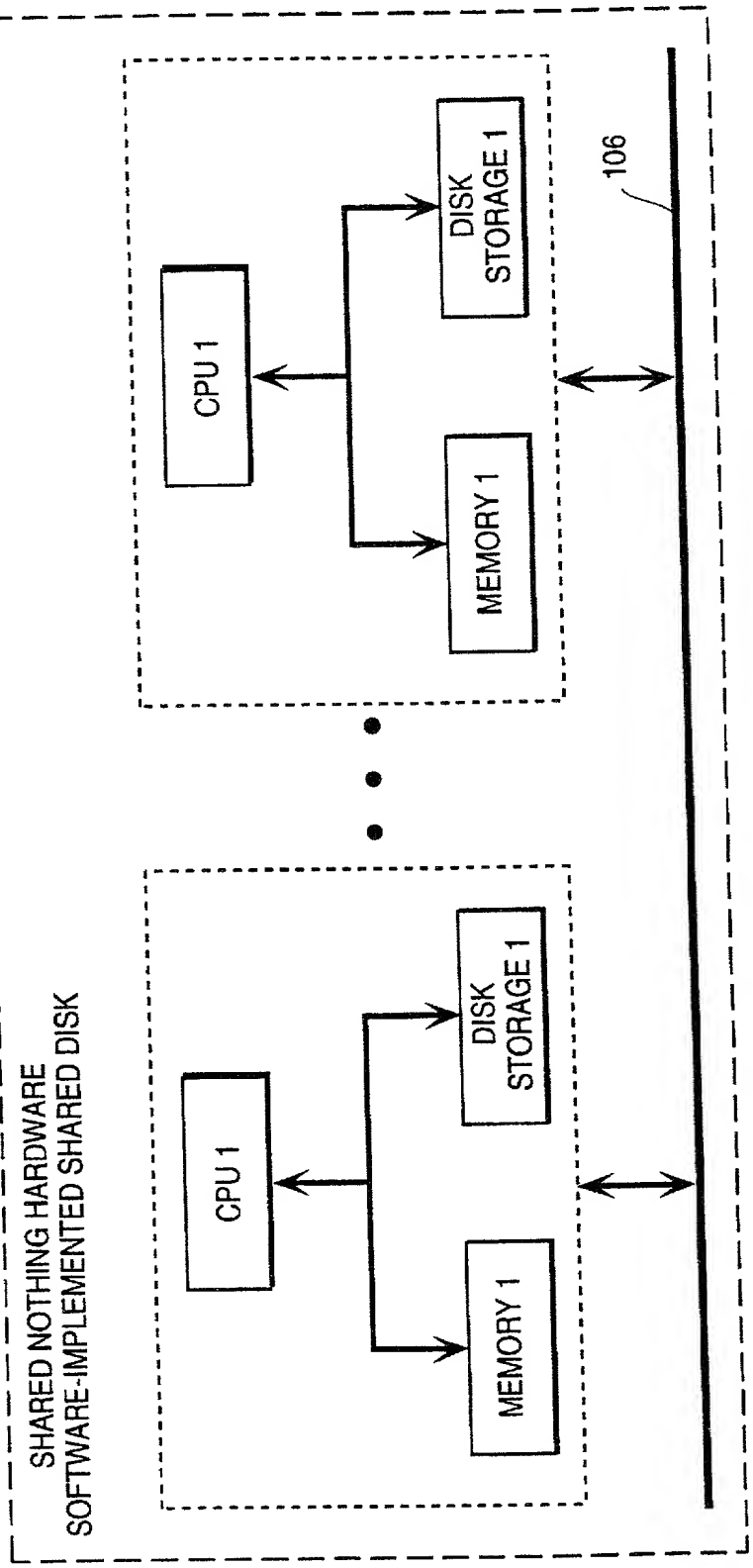

FIG. 1D (PRIOR ART)

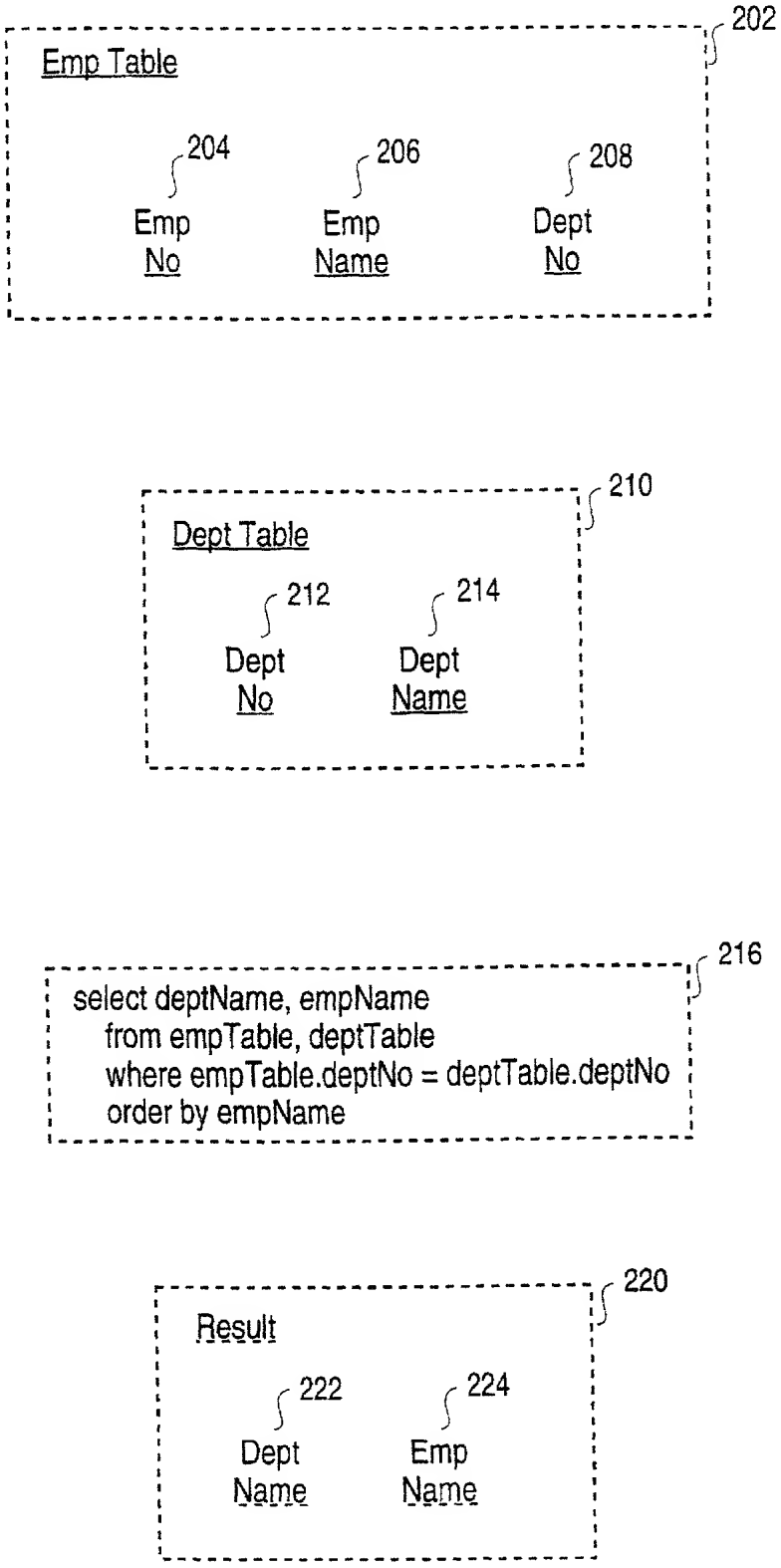
202
Emp Table
204
Emp No
206
Emp Name
208
Dept No
210
Dept Table
212
Dept No
214
Dept Name
216
select deptName, empName
from empTable, deptTable
where empTable.deptNo = deptTable.deptNo
order by empName
220
Result
222
Dept Name
224
Emp Name

FIG. 2

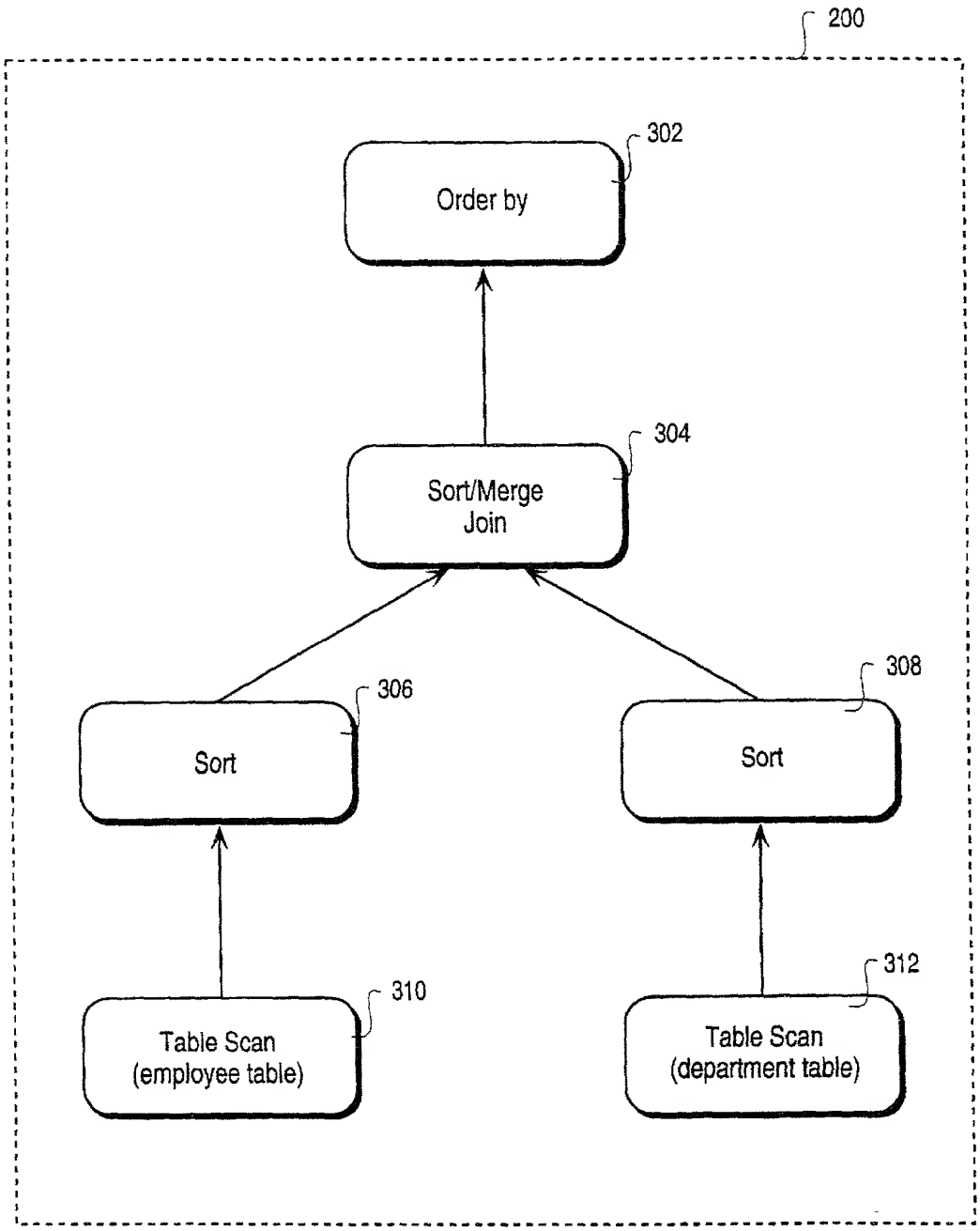
200
302
Order by
304
Sort/Merge Join
306
Sort
308
Sort
310
Table Scan (employee table)
312
Table Scan (department table)

FIG. 3A

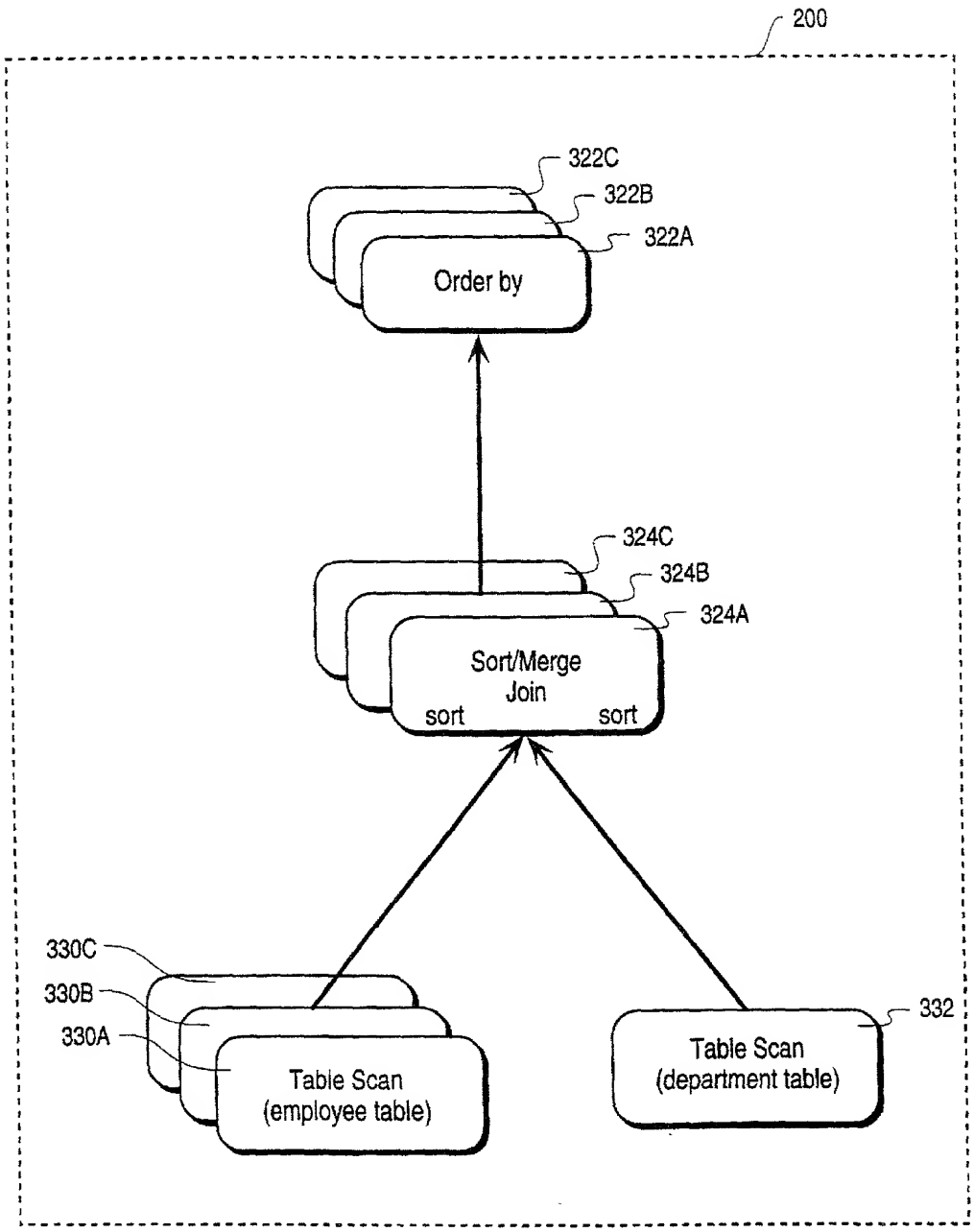
200
322C
322B
322A
Order by
324C
324B
324A
Sort/Merge
Join
sort
sort
330C
330B
330A
Table Scan
(employee table)
332
Table Scan
(department table)

FIG. 3B

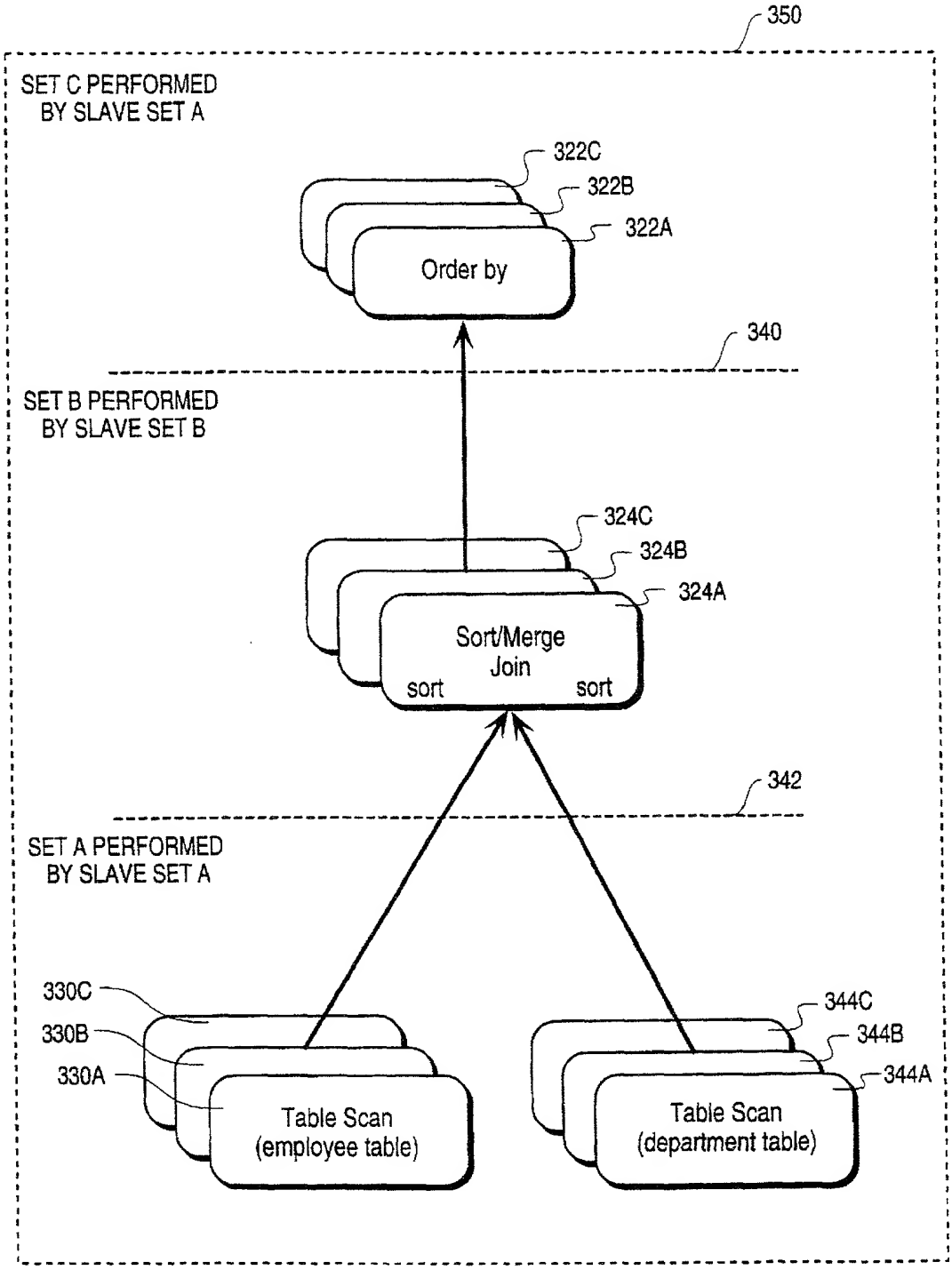
350
SET C PERFORMED BY SLAVE SET A
322C
322B
322A
Order by
340
SET B PERFORMED BY SLAVE SET B
324C
324B
324A
Sort/Merge Join
sort
sort
342
SET A PERFORMED BY SLAVE SET A
330C
330B
330A
Table Scan (employee table)
344C
344B
344A
Table Scan (department table)

FIG. 3C

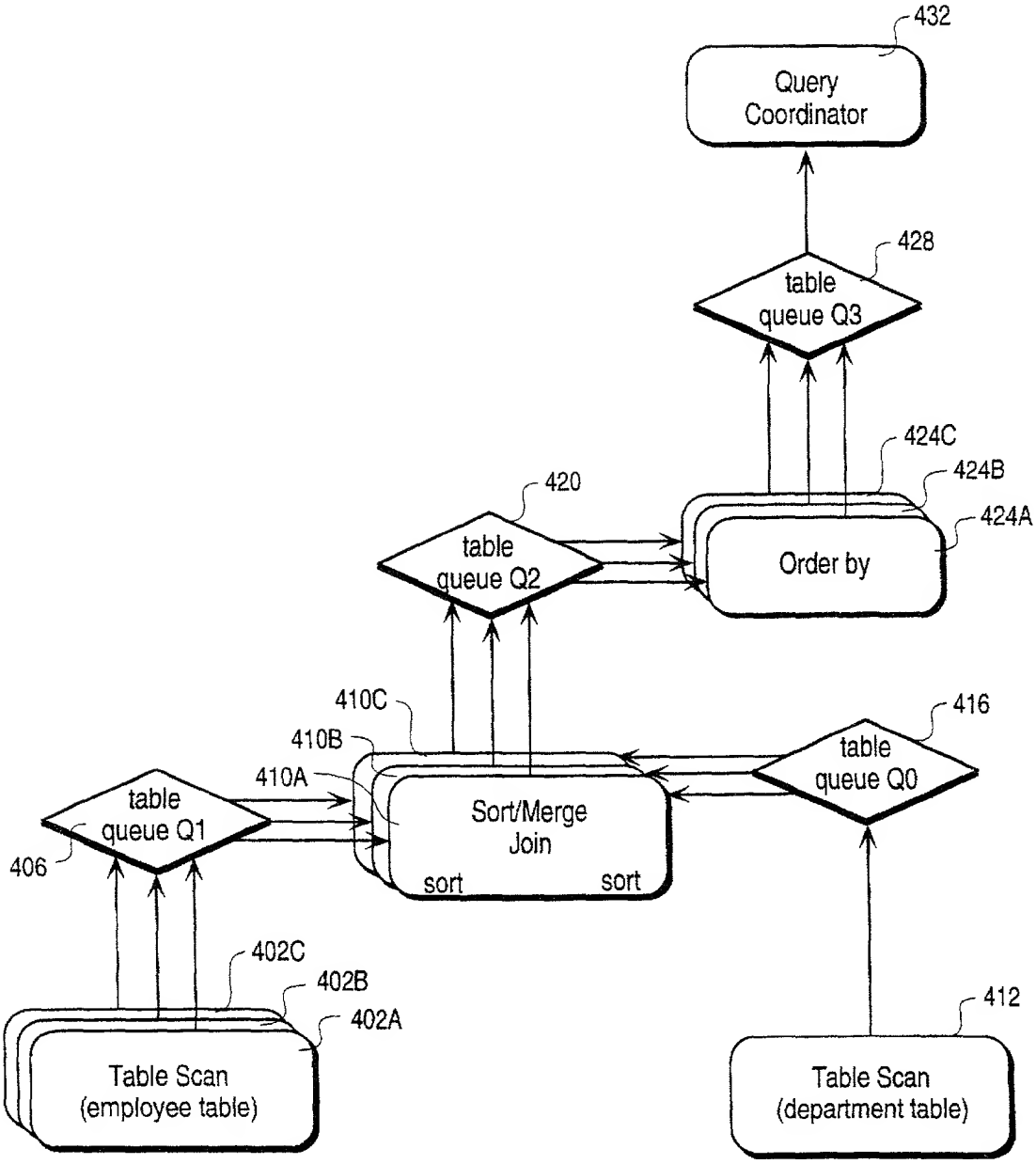
432
Query Coordinator
428
table queue Q3
424C
424B
424A
420
table queue Q2
Order by
410C
410B
410A
416
table queue Q0
table queue Q1
406
Sort/Merge Join
sort
sort
402C
402B
402A
Table Scan (employee table)
412
Table Scan (department table)

FIG. 4

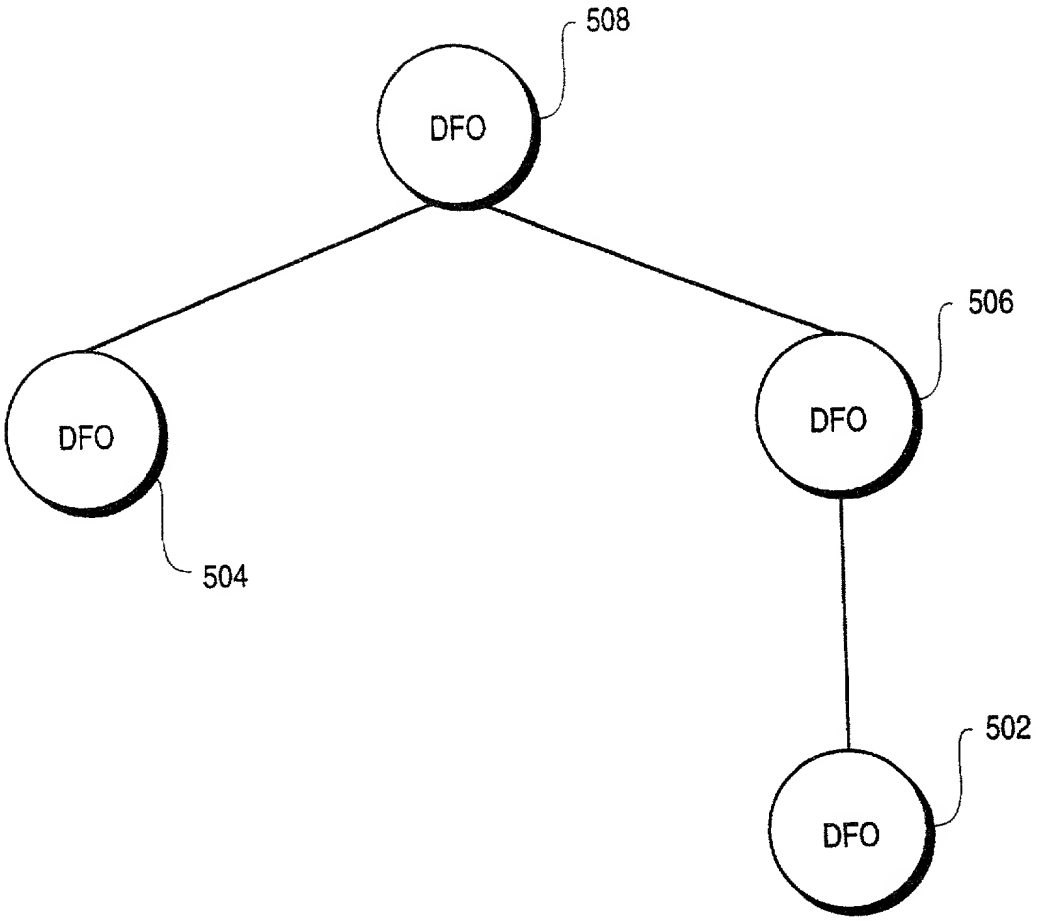

FIG. 5

DFO Type	Type of DFO (e.g., table scan, sort/merge join)
Pointer to Parent	Pointer to parent node
Pointer to Sibling	Pointer to next-node
Row Source	Row source equivilent for this node
Operation	
Table Queue Identifier	Identifies table node for this queue
Skipped Node Information	
Non-QC Node Infomation	
Pointer to First Child	Pointer to first child of parallelized node
Key Column Number	Number of key columns in input table queue
Partitioning Type	Parellelized node's partitioning type (e.g., hash, key range, rowid range, round robin)
Clumped Columns	Number of clumped colums with parent
Predicates	
Control Blocks	
Table Scan	Contains table scan information (e.g., table name and degree of parallelism)
Indexed Nested Loop Join	Contains information for an indexed nested loop join (e.g., right and left input table)
names)	
Sort/Merge join	Contains information for a sort/merge join (e.g., merge or outer join control flags)
Index Creation	Contains information for index creation (e.g., column list, index type, and storage parameters)

FIG. 6A

Pointer to Next-to-Execute
Table Queue Identifier
Rowid Table Number
SQL Statement Size
SQL Statement
Flags
Started
Ready
Close

Pointer to the next DFO to execute
Output table queue identifier
Number of tables partitioned by rowid
Size of the SQL statement representing DFO
SQL statement representing DFO

Slave sends "Started" message upon receipt
Slave sends "Ready" message when input consumed
Slave expects to be closed upon completion

FIG. 6B

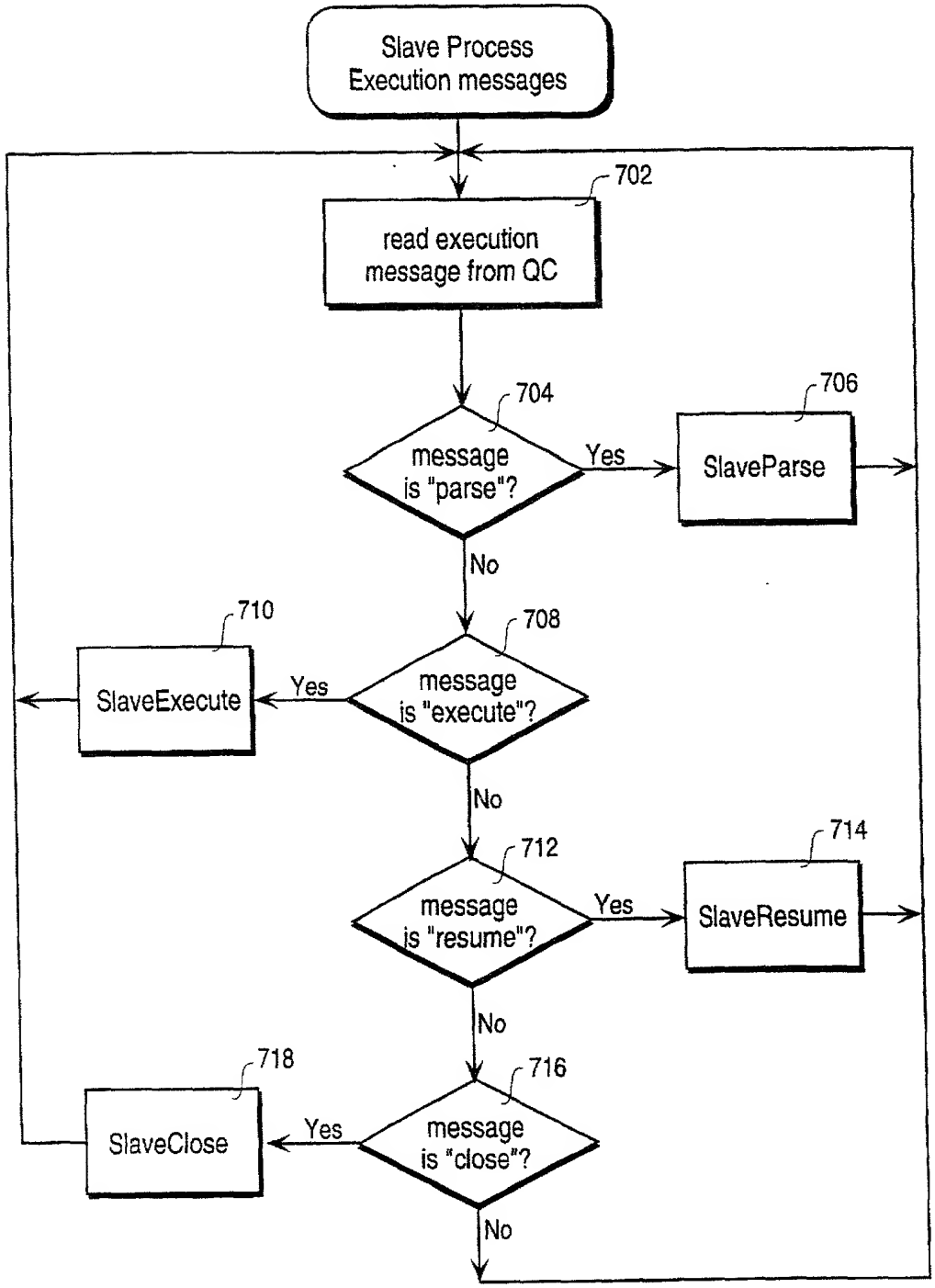
Slave Process Execution messages
702
read execution message from QC
704
message is "parse"?
Yes
706
SlaveParse
No
708
message is "execute"?
Yes
710
SlaveExecute
No
712
message is "resume"?
Yes
714
SlaveResume
No
716
message is "close"?
Yes
718
SlaveClose
No

FIG. 7A

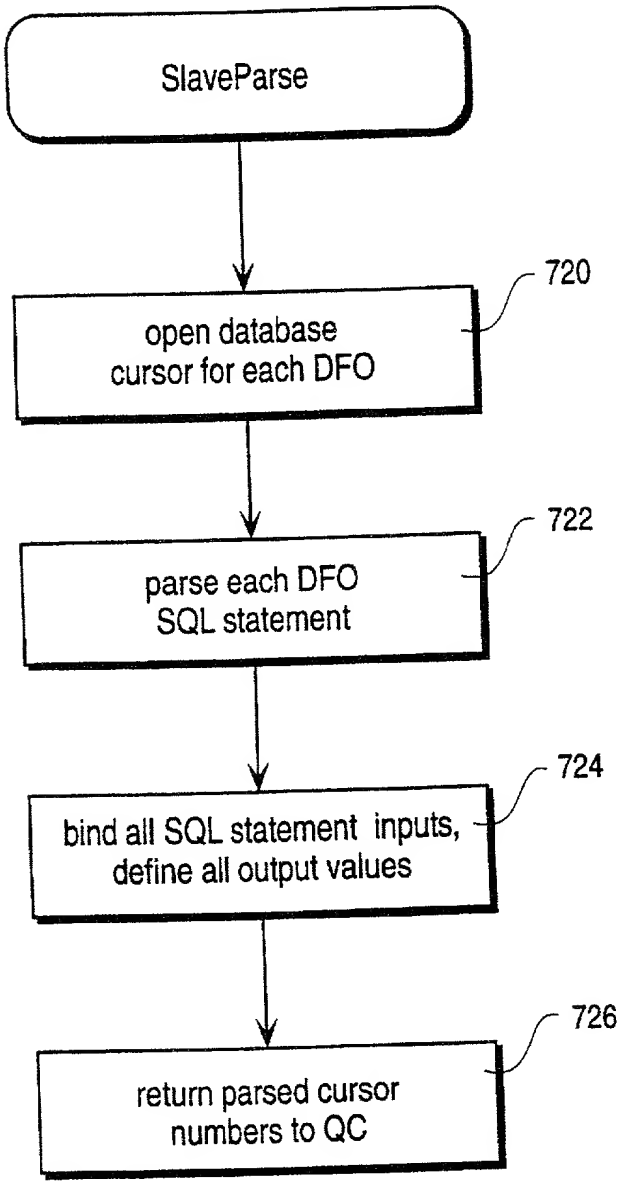
SlaveParse
open database cursor for each DFO
720
parse each DFO SQL statement
722
bind all SQL statement inputs, define all output values
724
return parsed cursor numbers to QC
726

FIG. 7B

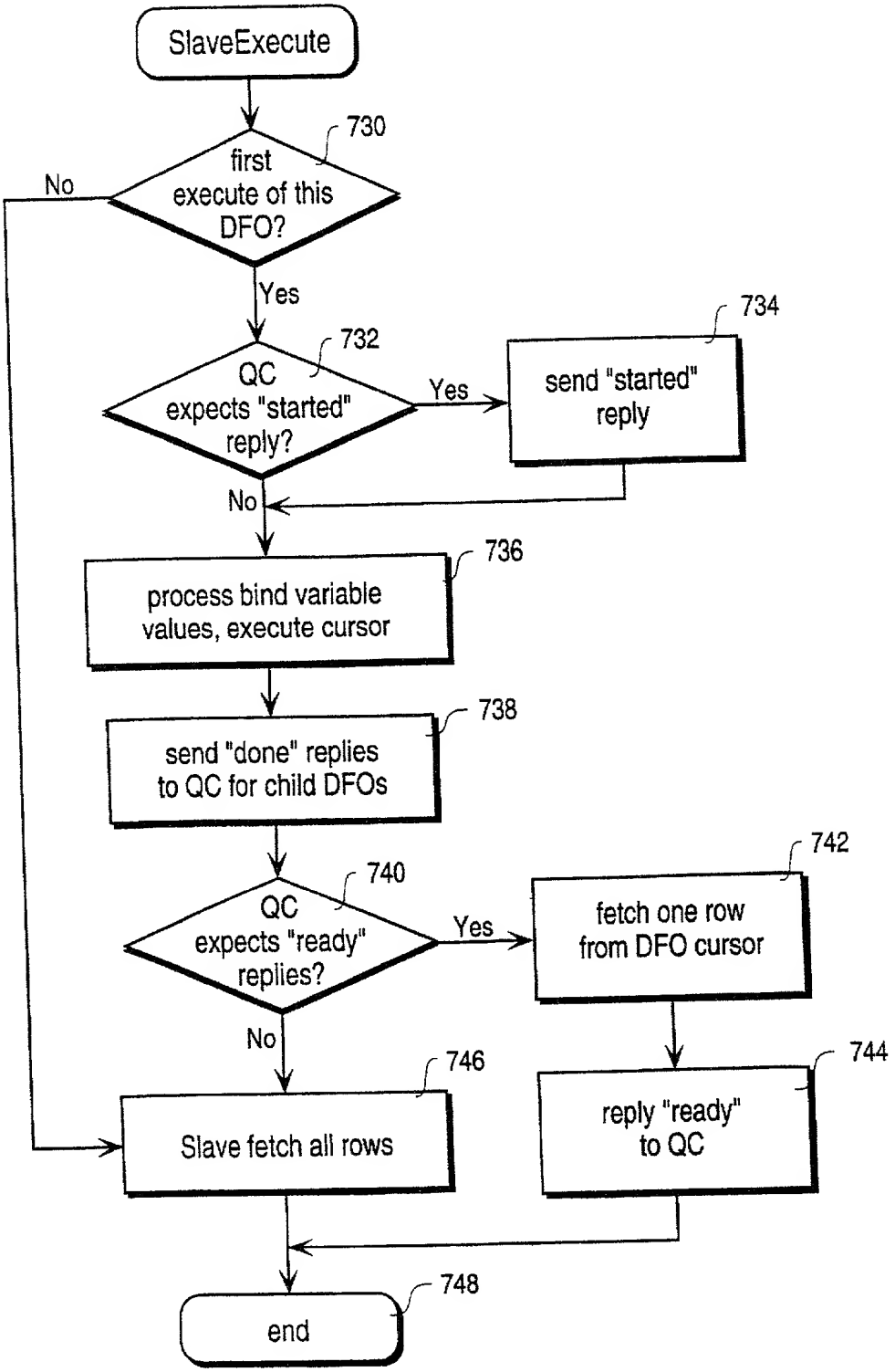
SlaveExecute
730
first execute of this DFO?
No
Yes
732
QC expects "started" reply?
Yes
734
send "started" reply
No
736
process bind variable values, execute cursor
738
send "done" replies to QC for child DFOs
740
QC expects "ready" replies?
Yes
742
fetch one row from DFO cursor
No
746
Slave fetch all rows
744
reply "ready" to QC
748
end

FIG. 7C

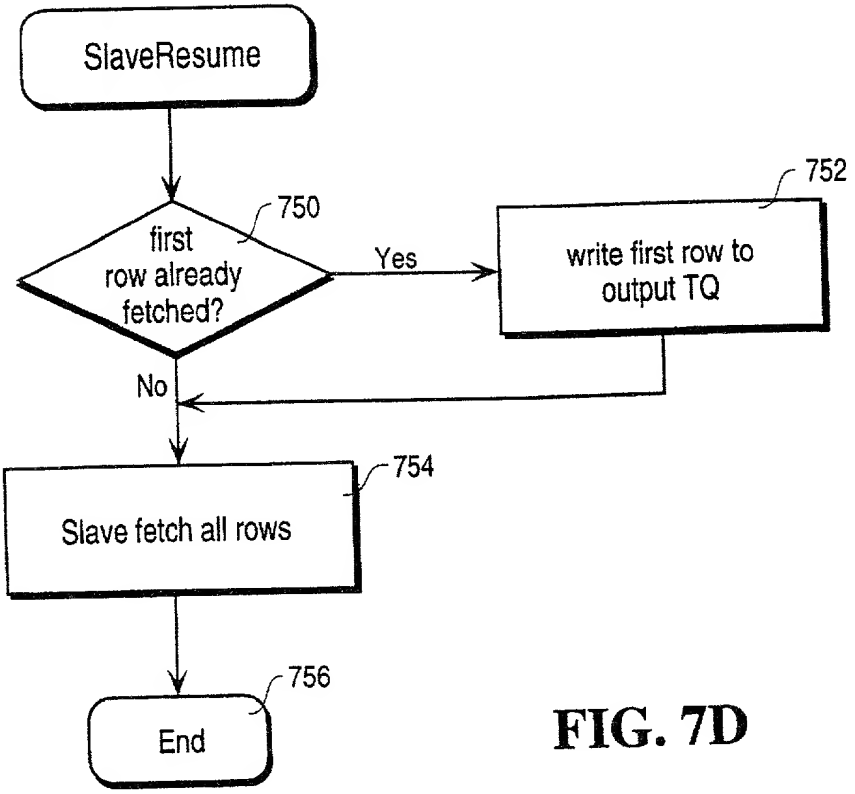
SlaveResume
750
first row already fetched?
Yes
752
write first row to output TQ
No
754
Slave fetch all rows
756
End

FIG. 7D

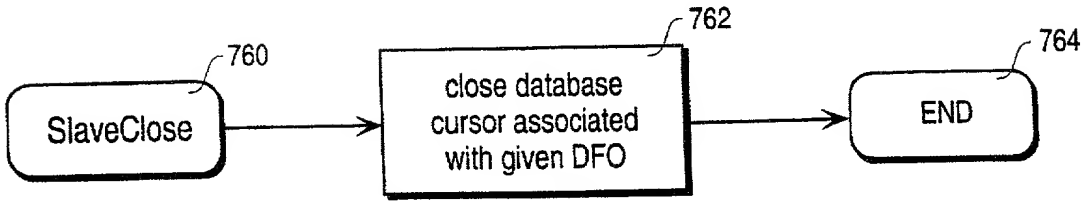
760
SlaveClose
762
close database cursor associated with given DFO
764
END

FIG. 7E

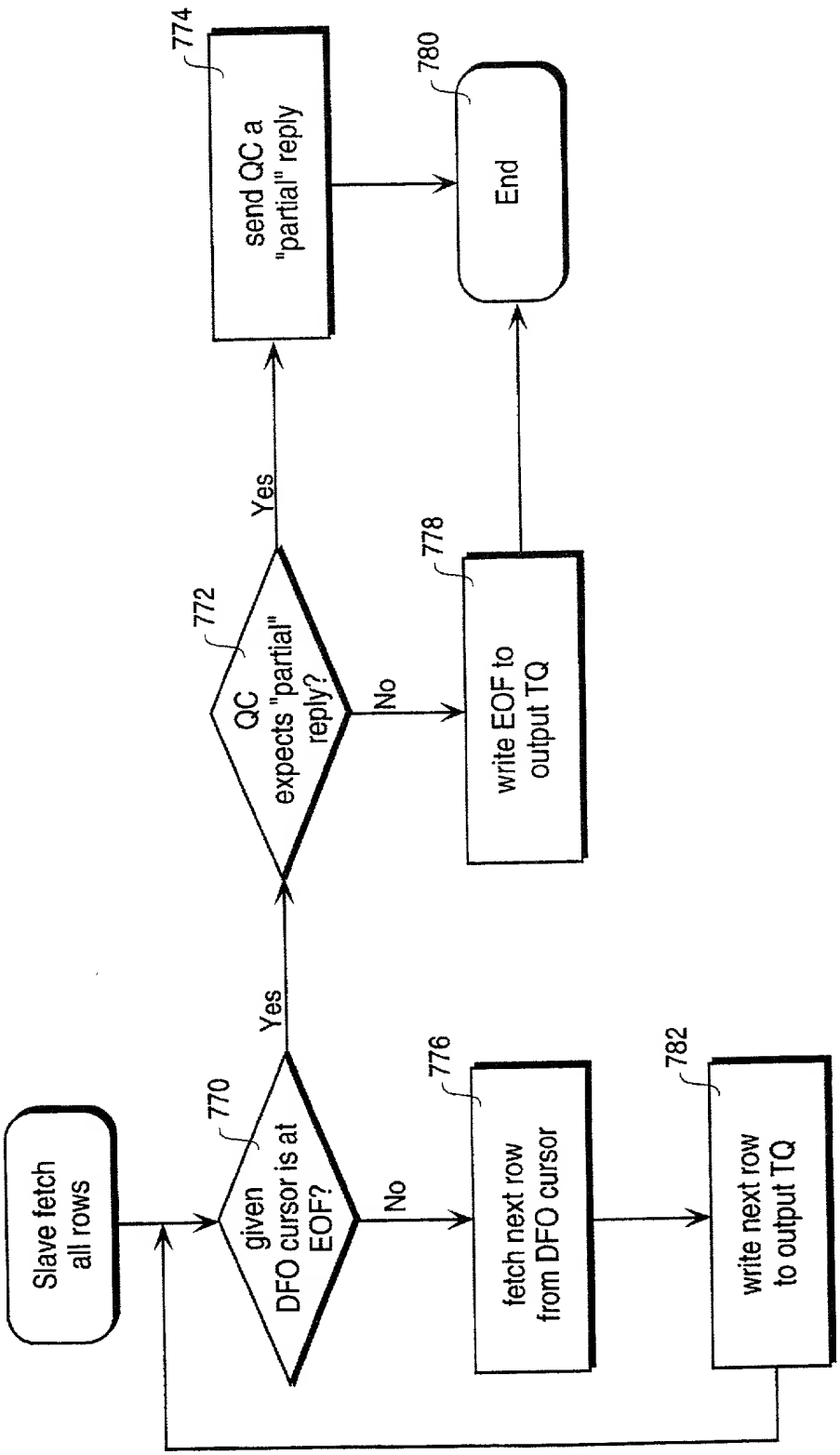
Slave fetch all rows
770
given DFO cursor is at EOF?
Yes
No
772
QC expects "partial" reply?
Yes
No
774
send QC a "partial" reply
776
fetch next row from DFO cursor
778
write EOF to output TQ
780
End
782
write next row to output TQ

FIG. 7F

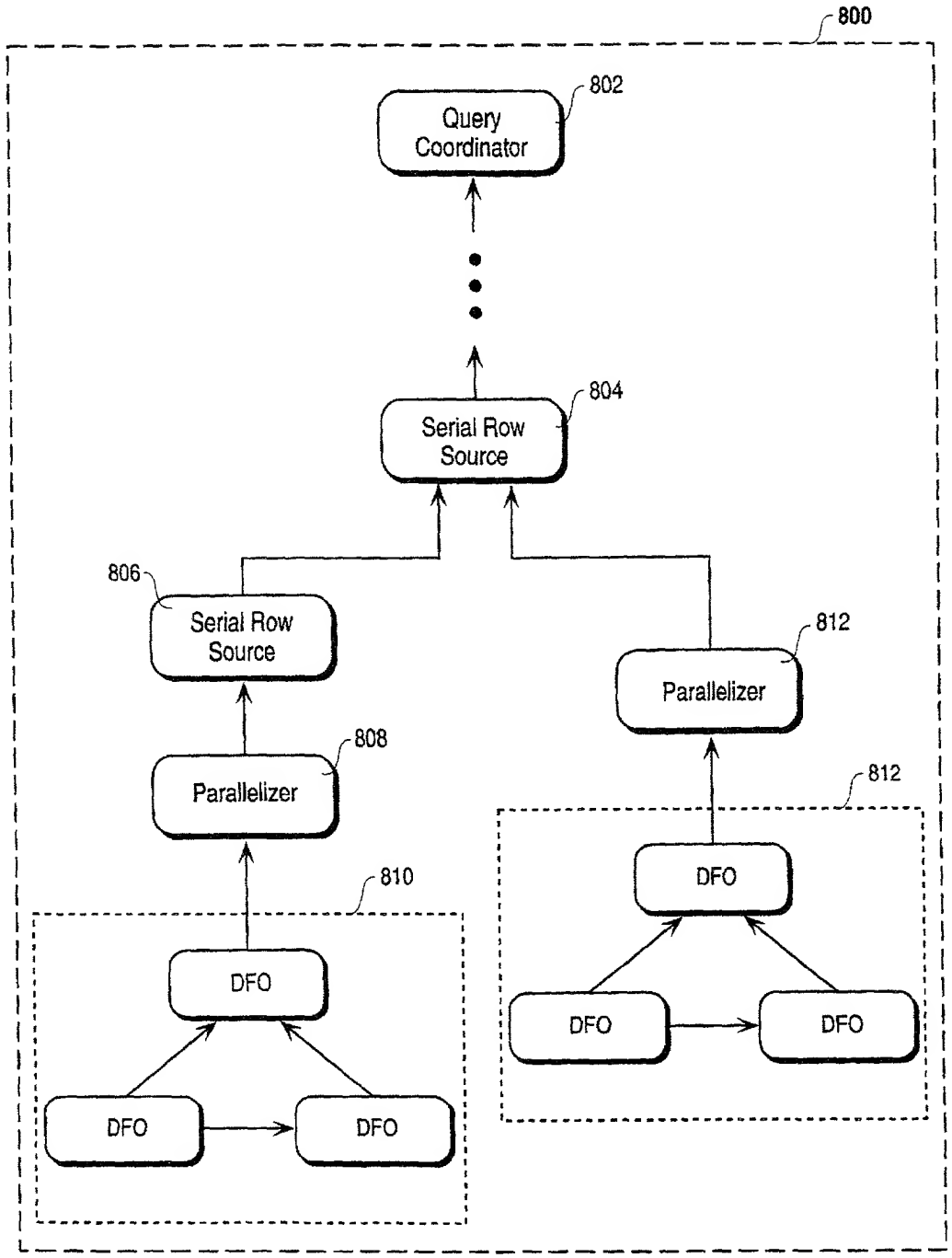
800
802
Query Coordinator
804
Serial Row Source
806
Serial Row Source
808
Parallelizer
810
DFO
DFO
DFO
812
Parallelizer
812
DFO
DFO
DFO

FIG. 8

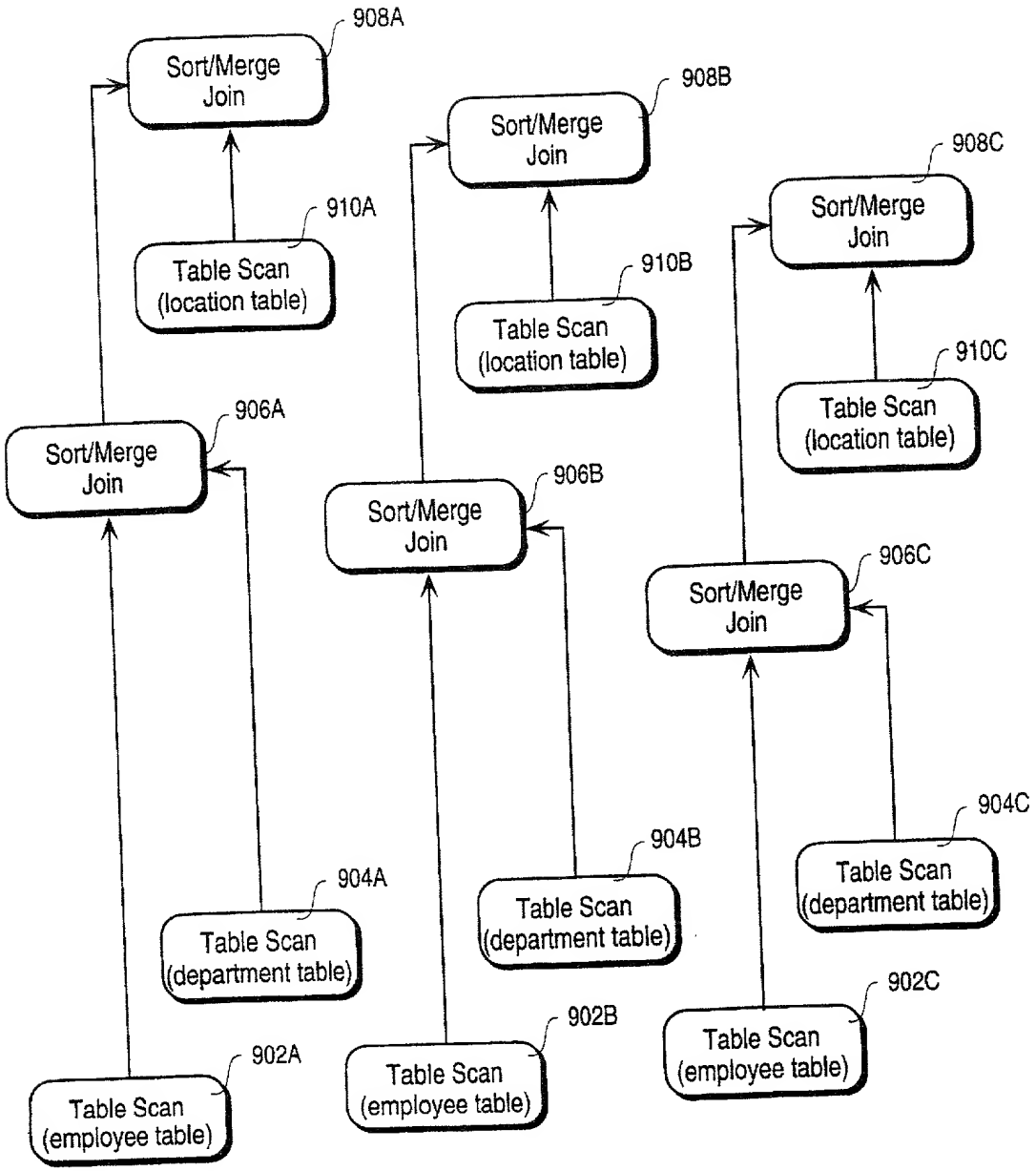
908A
Sort/Merge Join
908B
Sort/Merge Join
908C
Sort/Merge Join
910A
Table Scan (location table)
910B
Table Scan (location table)
910C
Table Scan (location table)
906A
Sort/Merge Join
906B
Sort/Merge Join
906C
Sort/Merge Join
904A
Table Scan (department table)
904B
Table Scan (department table)
904C
Table Scan (department table)
902A
Table Scan (employee table)
902B
Table Scan (employee table)
902C
Table Scan (employee table)

FIG. 9

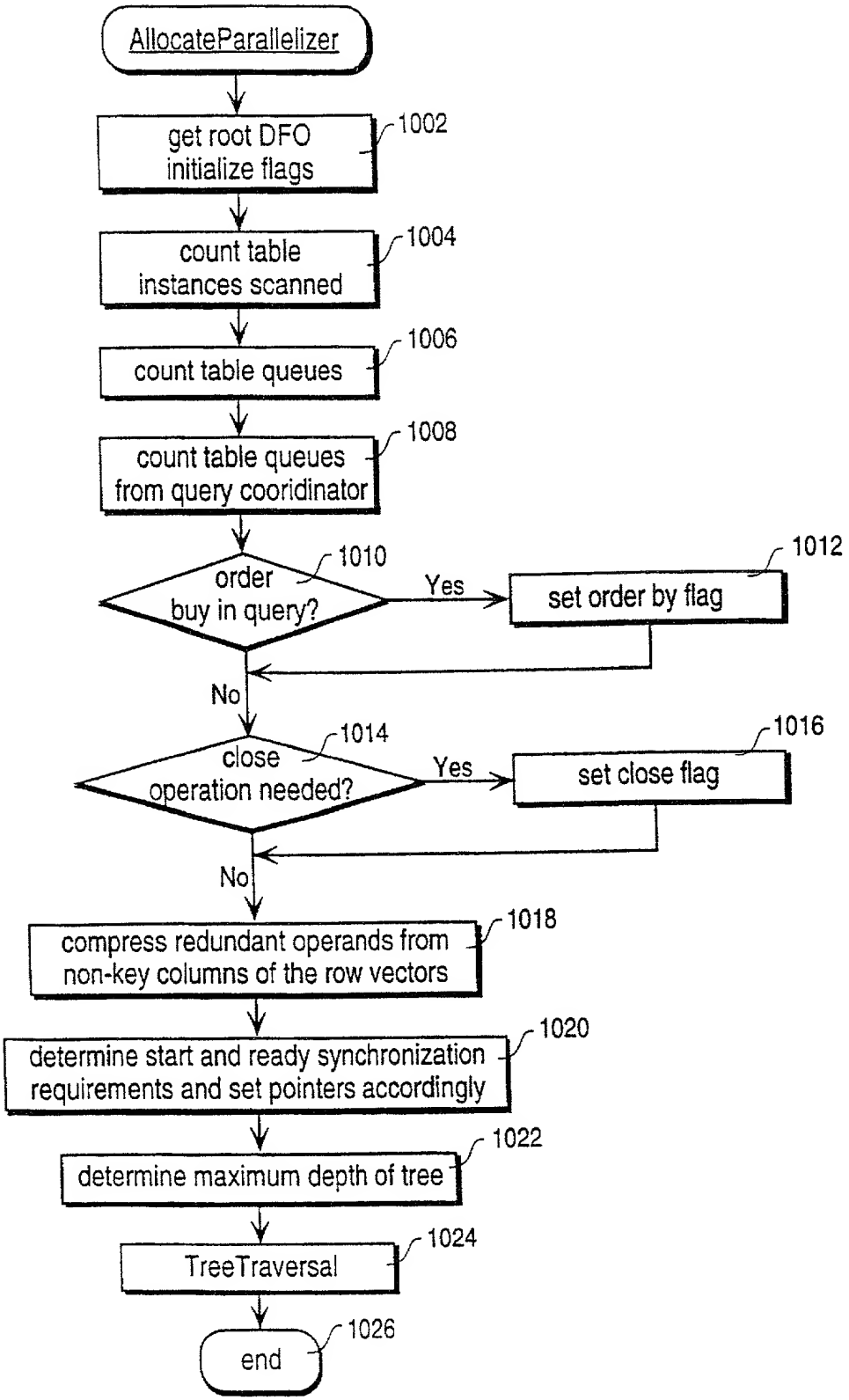
AllocateParallelizer
get root DFO initialize flags
1002
count table instances scanned
1004
count table queues
1006
count table queues from query cooridinator
1008
order buy in query?
1010
Yes
set order by flag
1012
No
close operation needed?
1014
Yes
set close flag
1016
No
compress redundant operands from non-key columns of the row vectors
1018
determine start and ready synchronization requirements and set pointers accordingly
1020
determine maximum depth of tree
1022
TreeTraversal
1024
end
1026

FIG. 10A

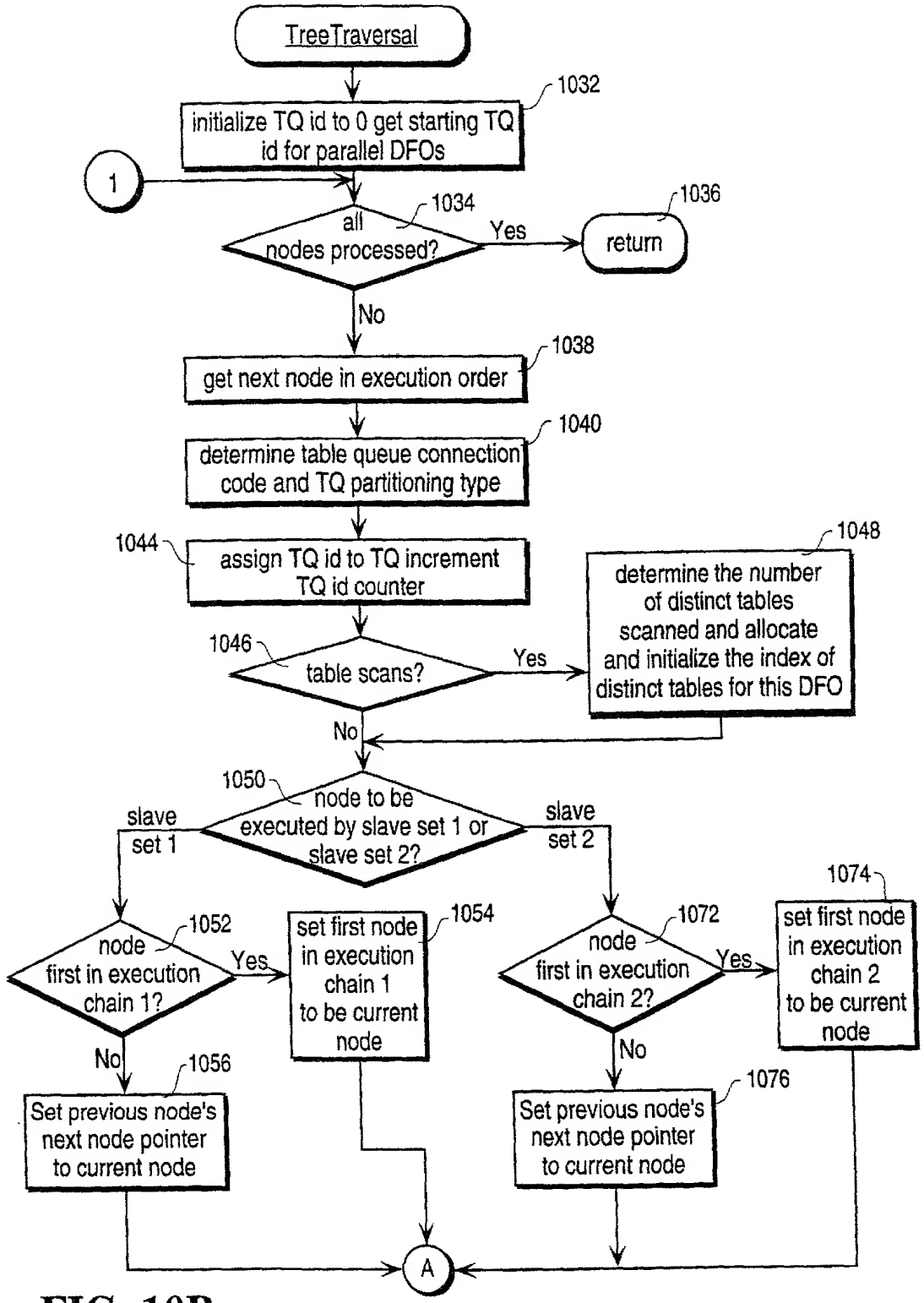
TreeTraversal
initialize TQ id to 0 get starting TQ id for parallel DFOs
1032
1
all nodes processed?
1034
Yes
return
1036
No
get next node in execution order
1038
determine table queue connection code and TQ partitioning type
1040
assign TQ id to TQ increment TQ id counter
1044
table scans?
1046
Yes
determine the number of distinct tables scanned and allocate and initialize the index of distinct tables for this DFO
1048
No
node to be executed by slave set 1 or slave set 2?
1050
slave set 1
slave set 2
node first in execution chain 1?
1052
Yes
set first node in execution chain 1 to be current node
1054
No
Set previous node's next node pointer to current node
1056
node first in execution chain 2?
1072
Yes
set first node in execution chain 2 to be current node
1074
No
Set previous node's next node pointer to current node
1076
A

FIG. 10B

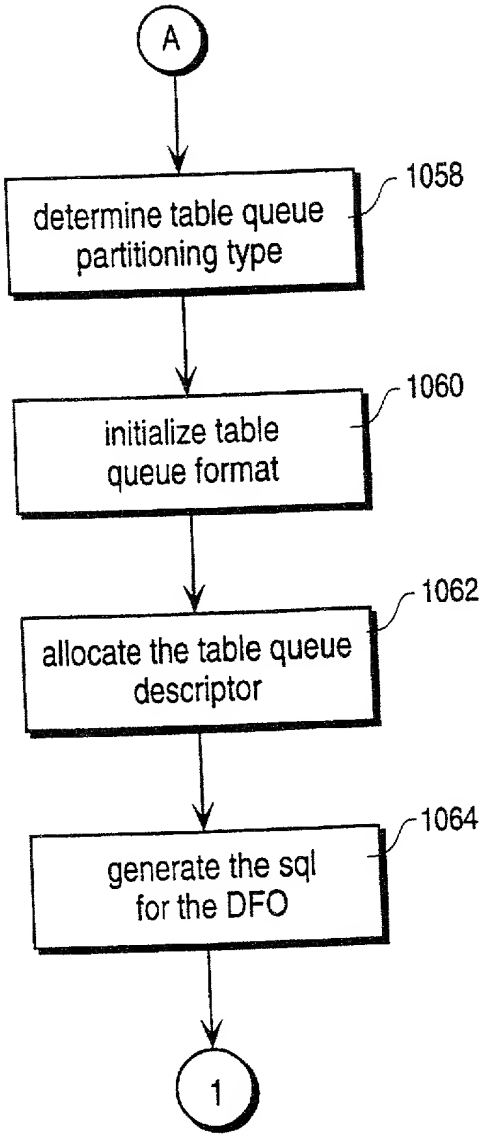
A
determine table queue partitioning type
1058
initialize table queue format
1060
allocate the table queue descriptor
1062
generate the sql for the DFO
1064
1

FIG. 10C

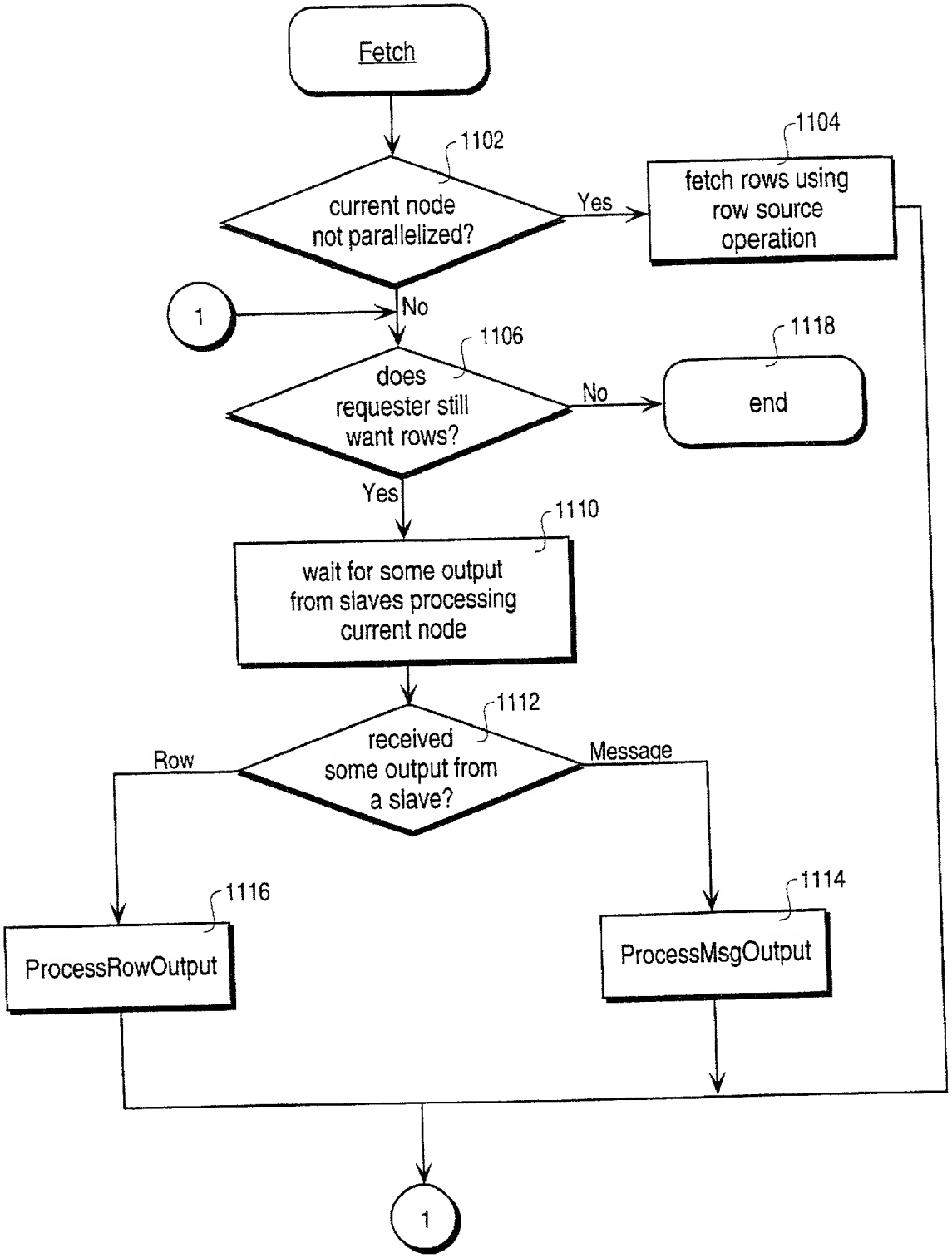
Fetch
1102
current node not parallelized?
Yes
1104
fetch rows using row source operation
1
No
1106
does requester still want rows?
No
1118
end
Yes
1110
wait for some output from slaves processing current node
1112
received some output from a slave?
Row
Message
1116
ProcessRowOutput
1114
ProcessMsgOutput
1

FIG. 11A

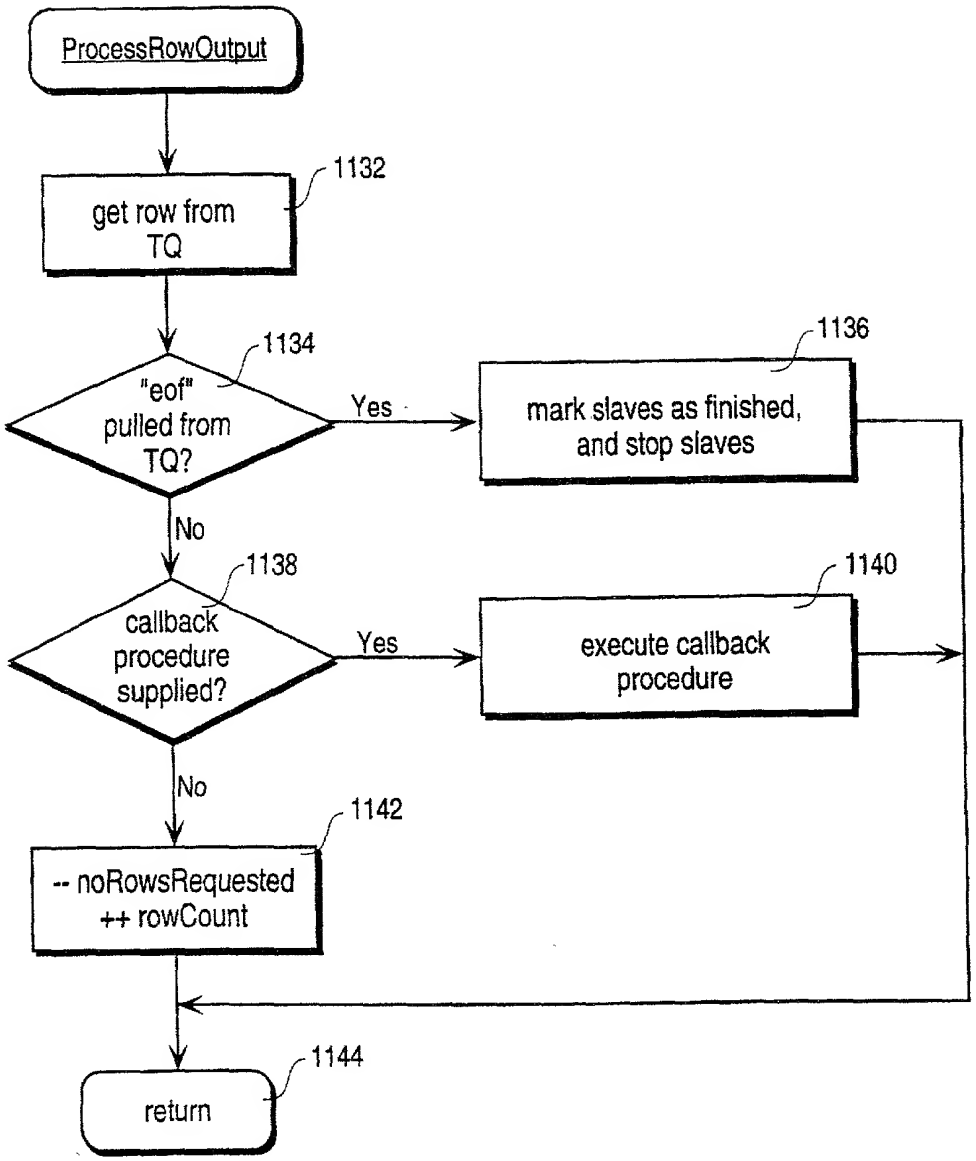
ProcessRowOutput
get row from TQ
1132
"eof" pulled from TQ?
1134
Yes
mark slaves as finished, and stop slaves
1136
No
callback procedure supplied?
1138
Yes
execute callback procedure
1140
No
-- noRowsRequested
++ rowCount
1142
return
1144

FIG. 11B

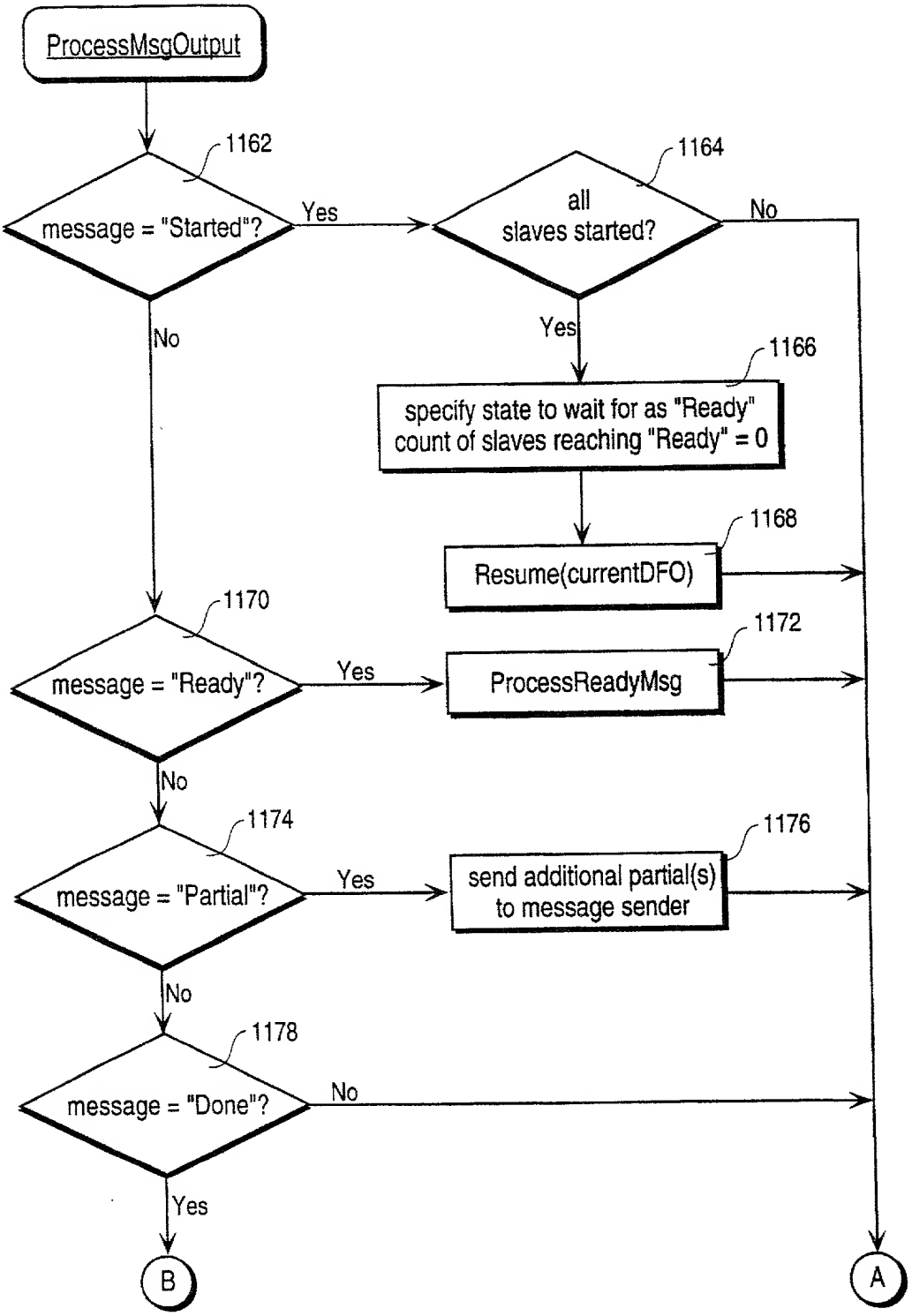
ProcessMsgOutput
1162
message = "Started"?
Yes
1164
all slaves started?
No
Yes
No
1166
specify state to wait for as "Ready"
count of slaves reaching "Ready" = 0
1168
Resume(currentDFO)
1170
message = "Ready"?
Yes
1172
ProcessReadyMsg
No
1174
message = "Partial"?
Yes
1176
send additional partial(s) to message sender
No
1178
message = "Done"?
No
Yes
B
A

FIG. 11C

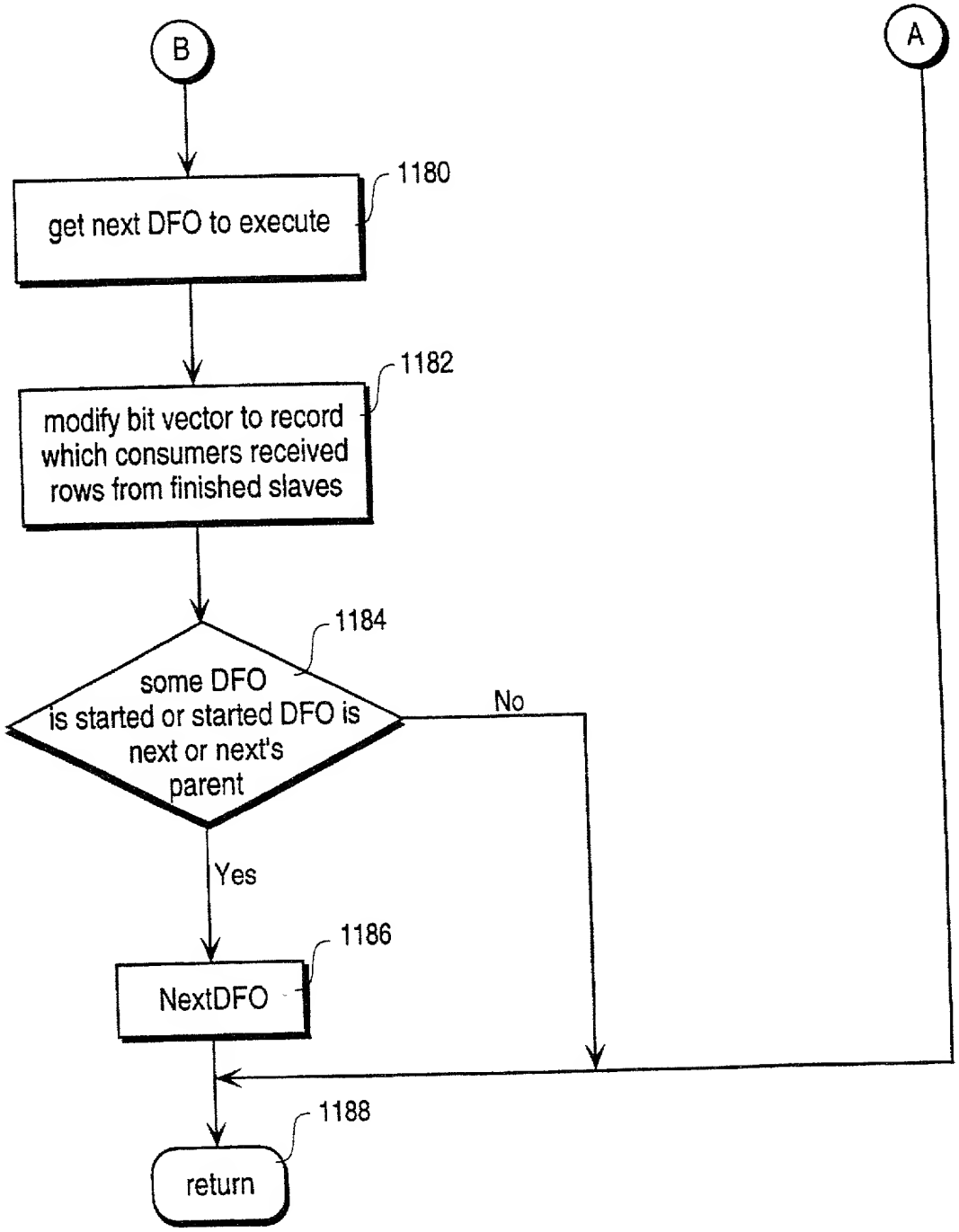
B
A
get next DFO to execute
1180
modify bit vector to record which consumers received rows from finished slaves
1182
some DFO is started or started DFO is next or next's parent
1184
No
Yes
NextDFO
1186
return
1188

FIG. 11D

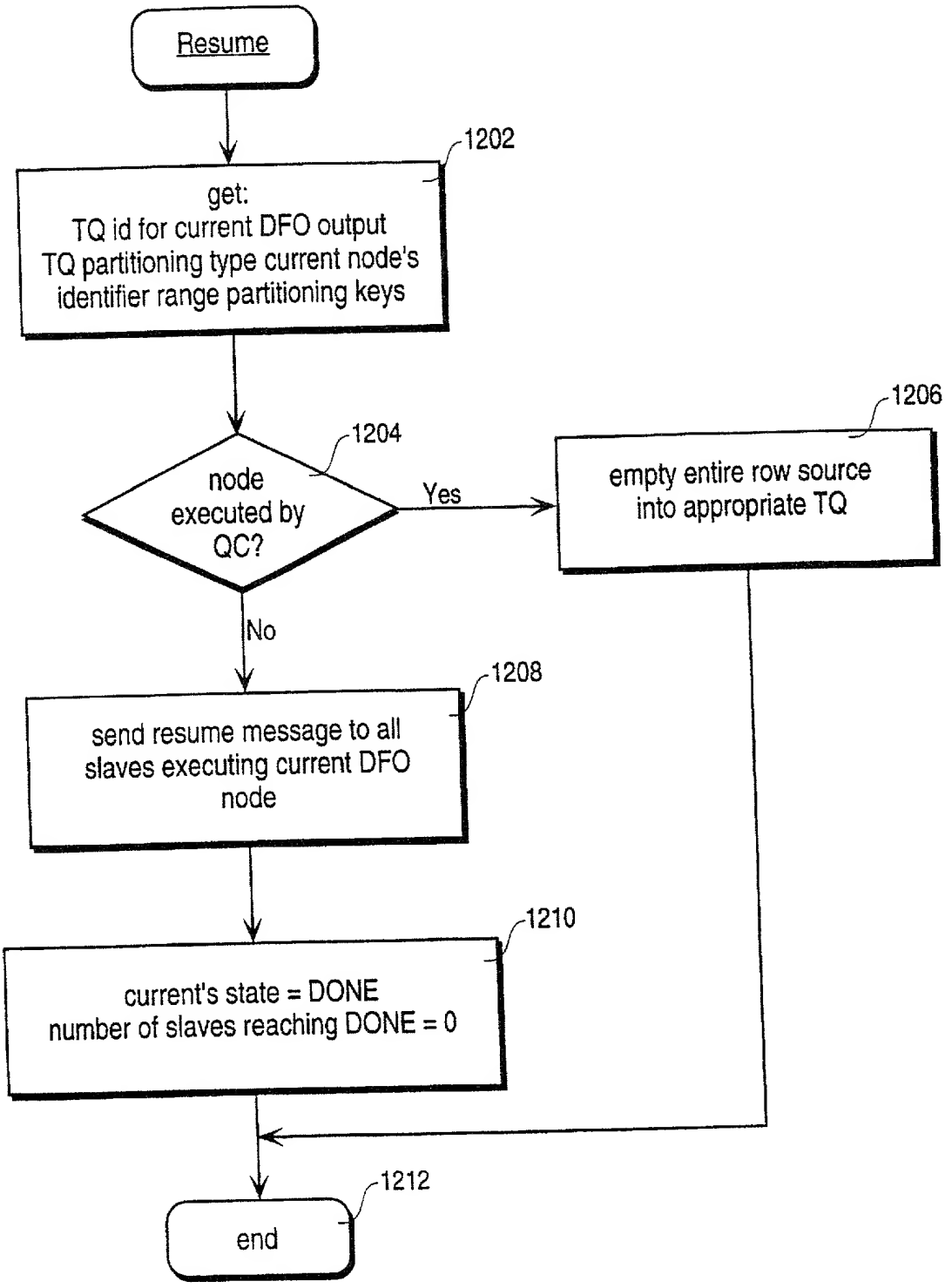
Resume
1202
get:
TQ id for current DFO output
TQ partitioning type current node's
identifier range partitioning keys
1204
node
executed by
QC?
Yes
1206
empty entire row source
into appropriate TQ
No
1208
send resume message to all
slaves executing current DFO
node
1210
current's state = DONE
number of slaves reaching DONE = 0
1212
end

FIG. 12

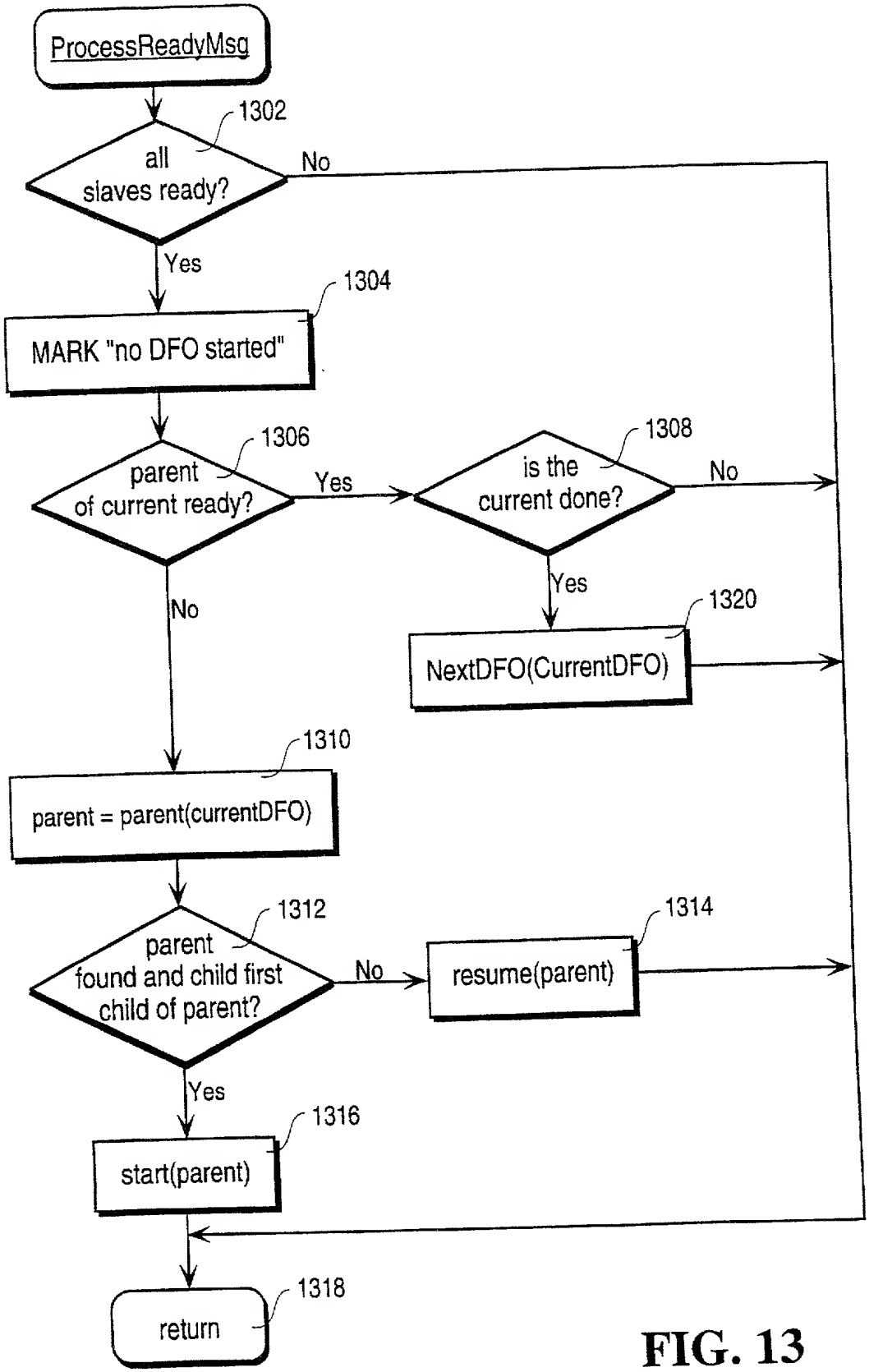
ProcessReadyMsg
1302
all slaves ready?
No
Yes
1304
MARK "no DFO started"
1306
parent of current ready?
Yes
1308
is the current done?
No
Yes
1320
NextDFO(CurrentDFO)
No
1310
parent = parent(currentDFO)
1312
parent found and child first child of parent?
No
1314
resume(parent)
Yes
1316
start(parent)
1318
return

FIG. 13

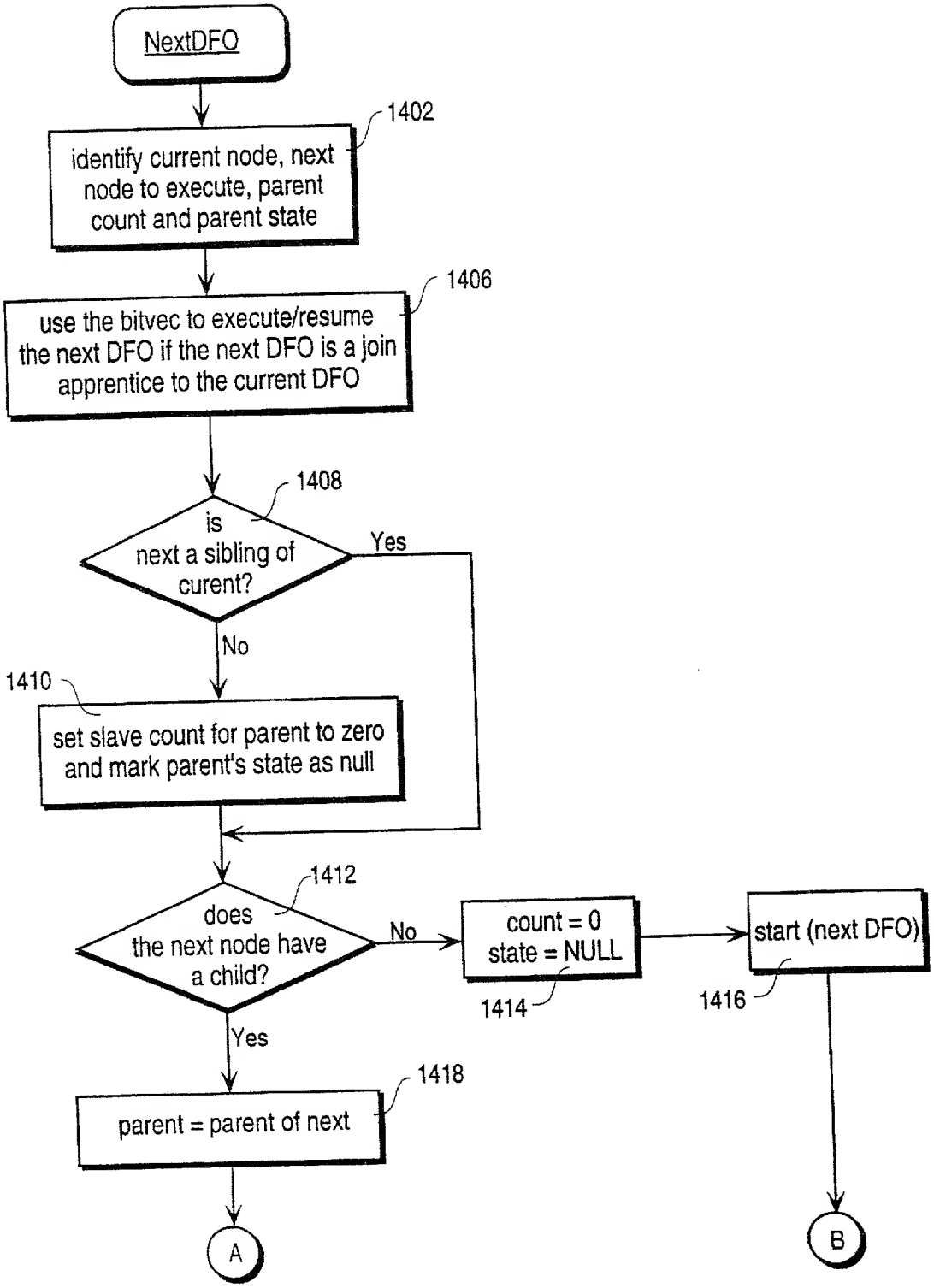
NextDFO
1402
identify current node, next node to execute, parent count and parent state
1406
use the bitvec to execute/resume the next DFO if the next DFO is a join apprentice to the current DFO
1408
is next a sibling of curent?
Yes
No
1410
set slave count for parent to zero and mark parent's state as null
1412
does the next node have a child?
No
count = 0 state = NULL
1414
start (next DFO)
1416
Yes
1418
parent = parent of next
A
B

FIG. 14A

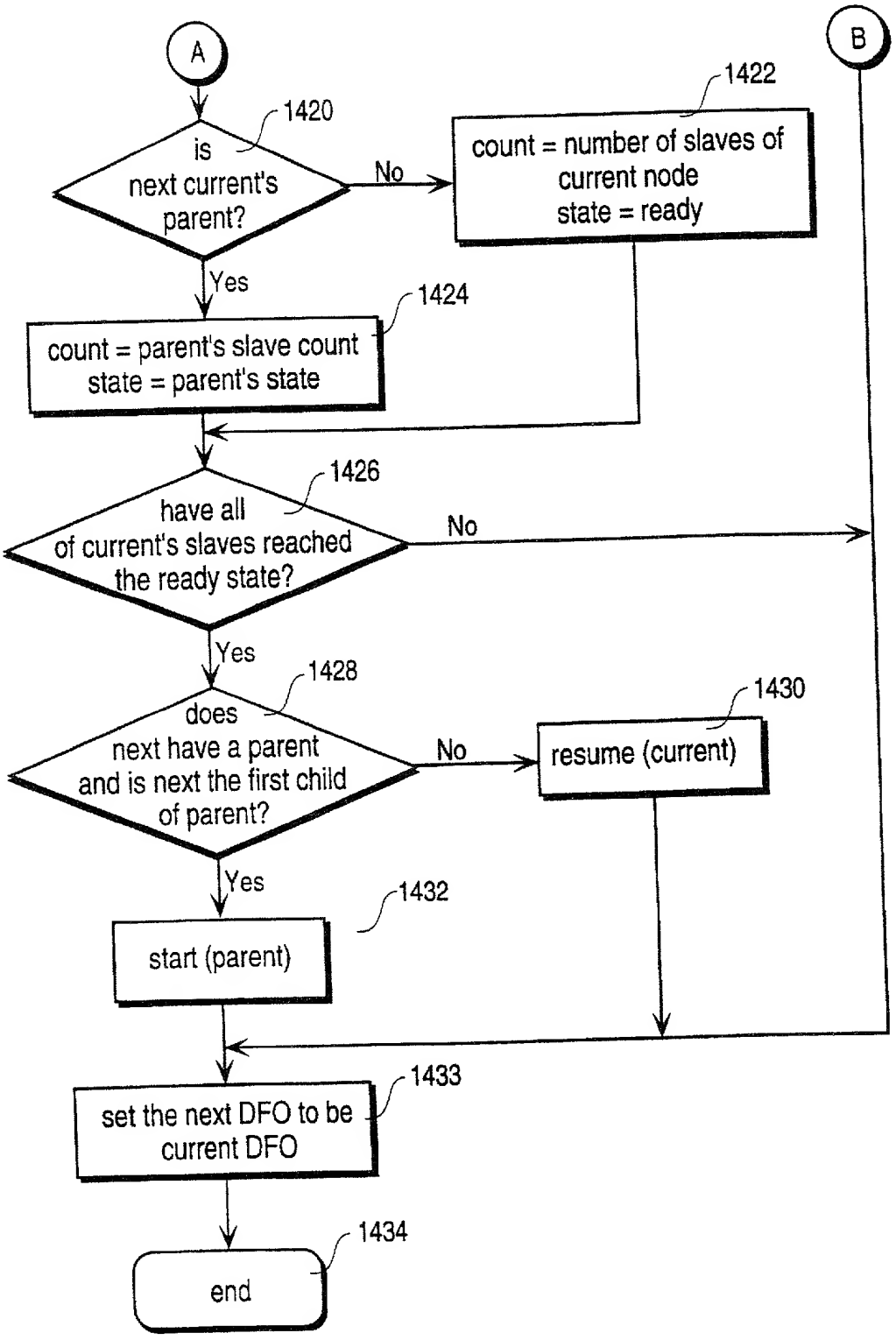
A
B
1420
is next current's parent?
No
1422
count = number of slaves of current node state = ready
Yes
1424
count = parent's slave count state = parent's state
1426
have all of current's slaves reached the ready state?
No
Yes
1428
does next have a parent and is next the first child of parent?
No
1430
resume (current)
Yes
1432
start (parent)
1433
set the next DFO to be current DFO
1434
end

FIG. 14B

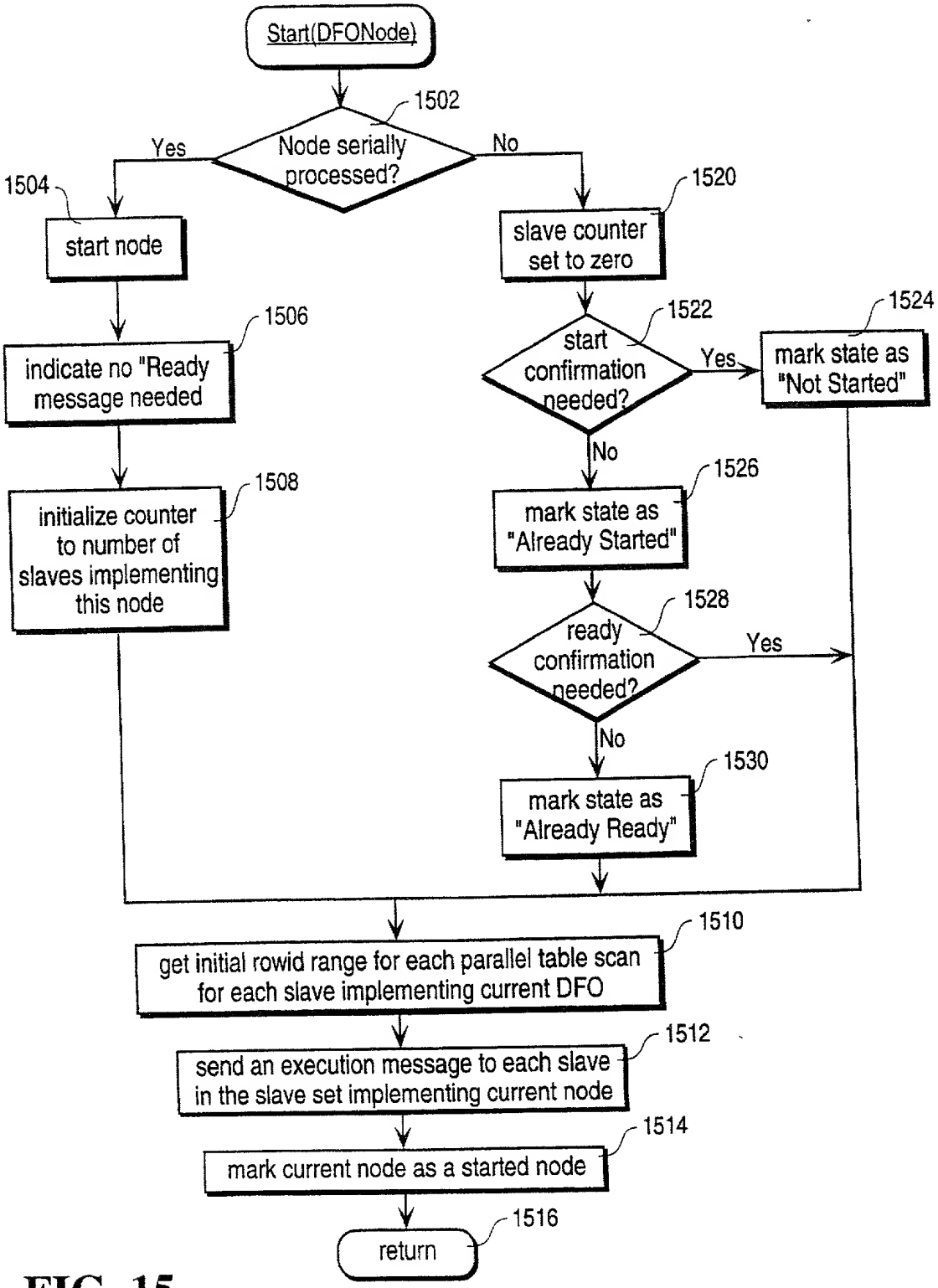
Start(DFONode)
1502
Node serially processed?
Yes
No
1504
start node
1506
indicate no "Ready message needed
1508
initialize counter to number of slaves implementing this node
1520
slave counter set to zero
1522
start confirmation needed?
Yes
1524
mark state as "Not Started"
No
1526
mark state as "Already Started"
1528
ready confirmation needed?
Yes
No
1530
mark state as "Already Ready"
1510
get initial rowid range for each parallel table scan for each slave implementing current DFO
1512
send an execution message to each slave in the slave set implementing current node
1514
mark current node as a started node
1516
return

FIG. 15

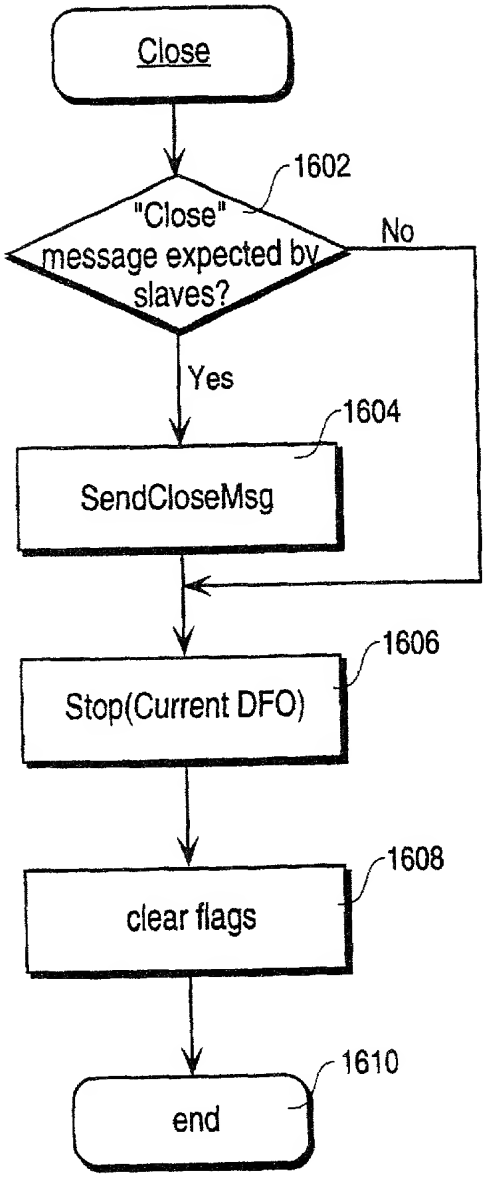
Close
1602
"Close" message expected by slaves?
No
Yes
1604
SendCloseMsg
1606
Stop(Current DFO)
1608
clear flags
1610
end

FIG. 16

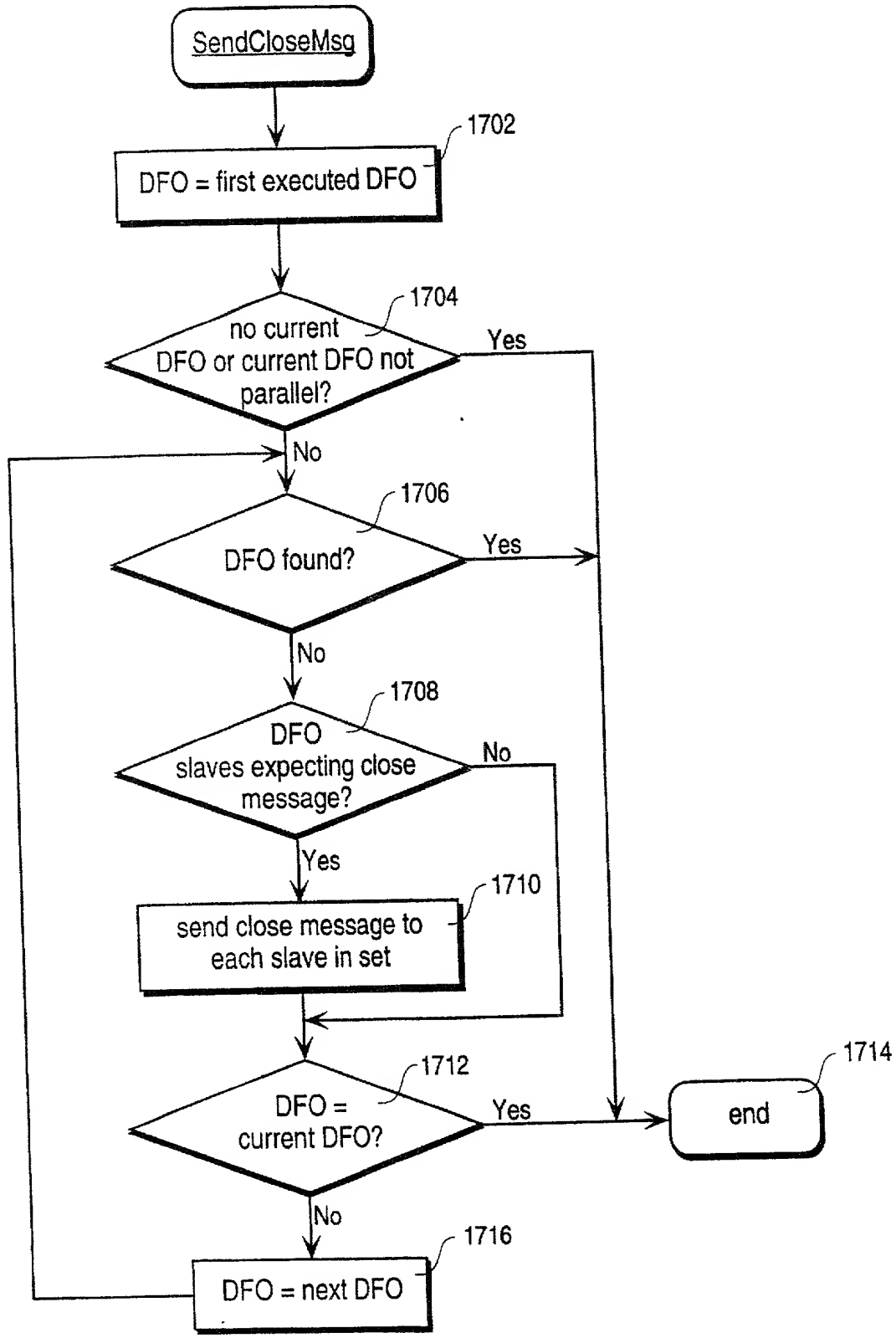
SendCloseMsg
1702
DFO = first executed DFO
1704
no current DFO or current DFO not parallel?
Yes
No
1706
DFO found?
Yes
No
1708
DFO slaves expecting close message?
No
Yes
1710
send close message to each slave in set
1712
DFO = current DFO?
Yes
1714
end
No
1716
DFO = next DFO

FIG. 17

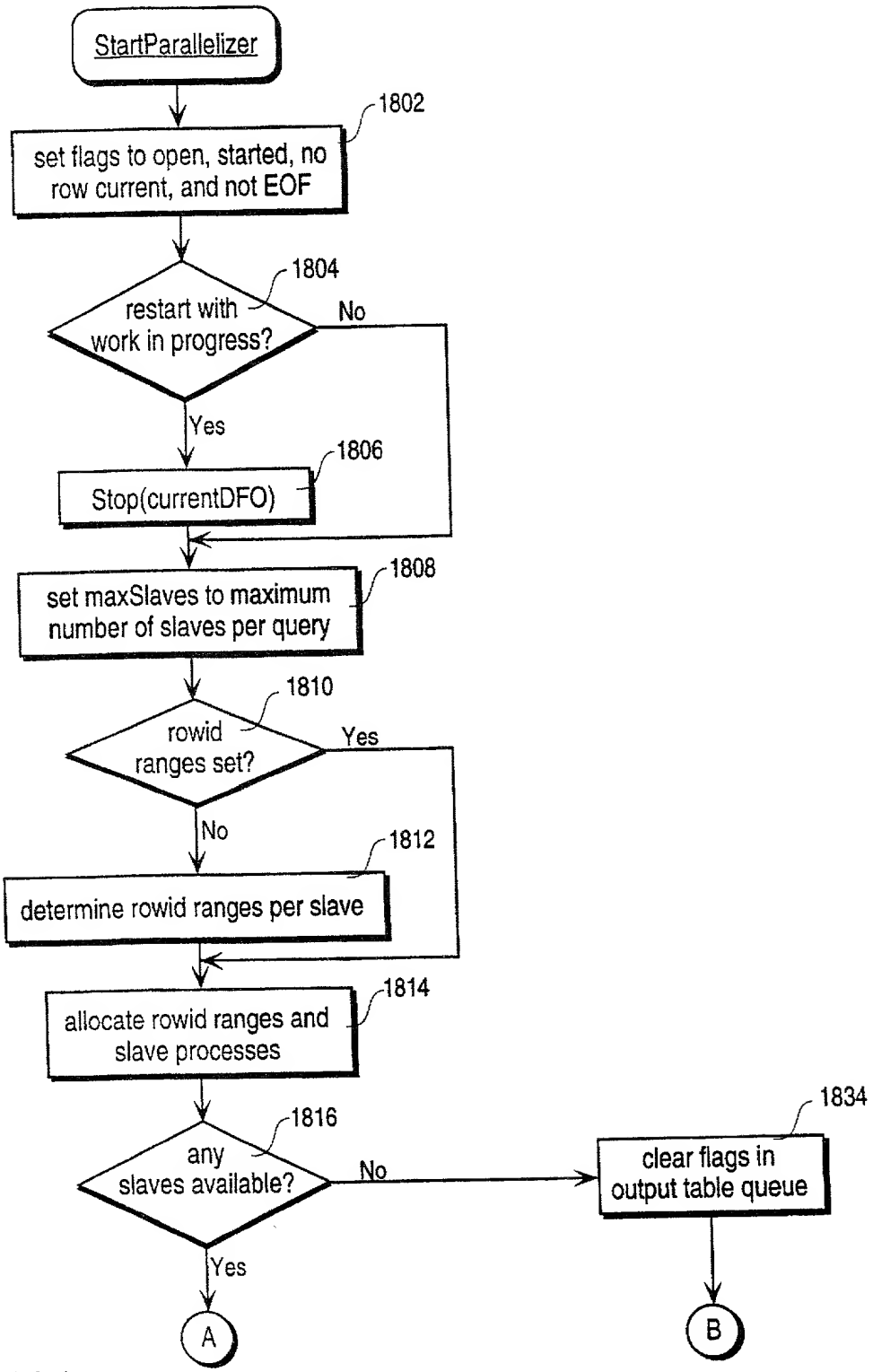
StartParallelizer
1802
set flags to open, started, no row current, and not EOF
1804
restart with work in progress?
No
Yes
1806
Stop(currentDFO)
1808
set maxSlaves to maximum number of slaves per query
1810
rowid ranges set?
Yes
No
1812
determine rowid ranges per slave
1814
allocate rowid ranges and slave processes
1816
any slaves available?
No
Yes
1834
clear flags in output table queue
A
B

FIG. 18A

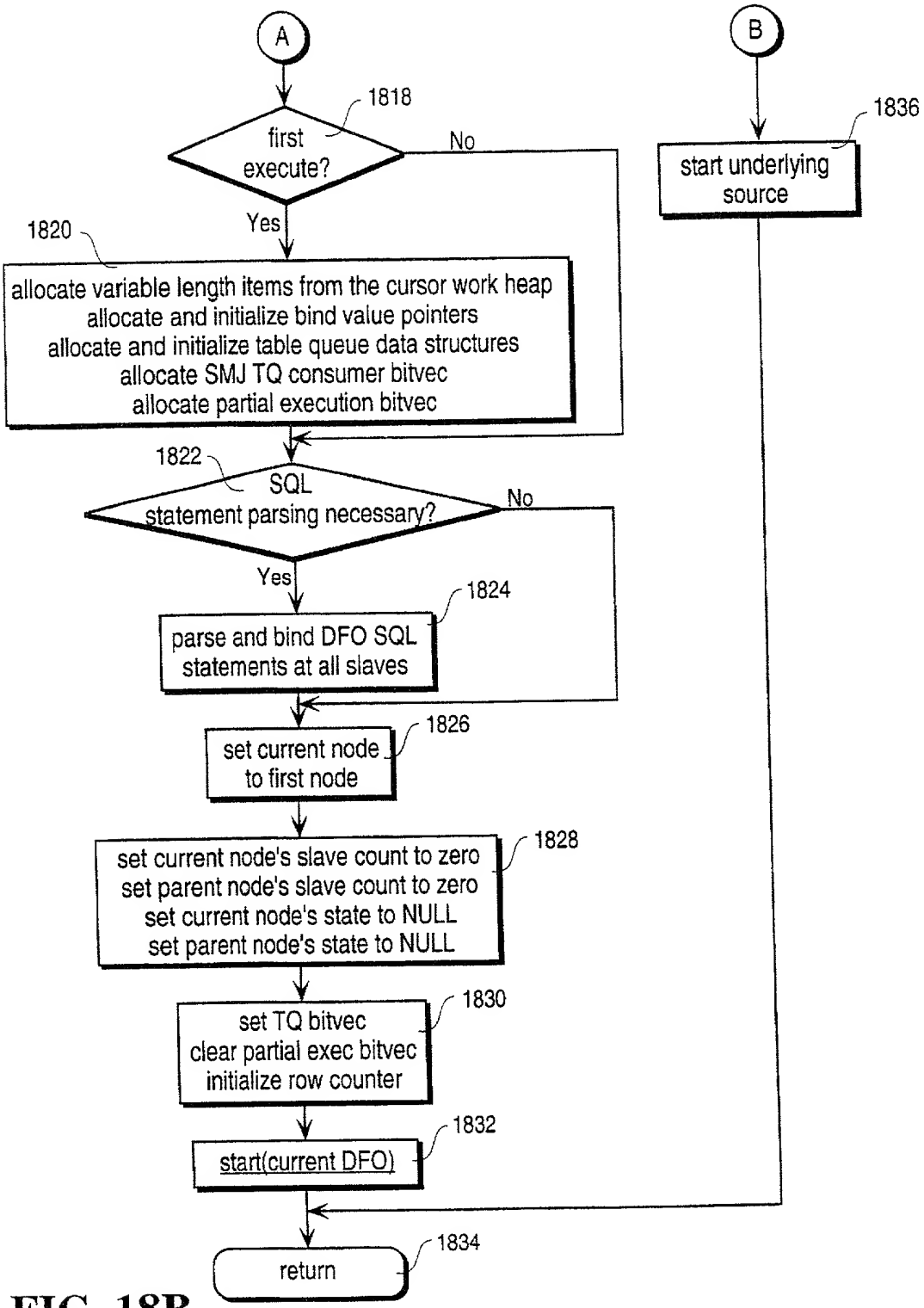
A
B
1818
first execute?
No
Yes
1820
allocate variable length items from the cursor work heap
allocate and initialize bind value pointers
allocate and initialize table queue data structures
allocate SMJ TQ consumer bitvec
allocate partial execution bitvec
1822
SQL statement parsing necessary?
No
Yes
1824
parse and bind DFO SQL statements at all slaves
1826
set current node to first node
1828
set current node's slave count to zero
set parent node's slave count to zero
set current node's state to NULL
set parent node's state to NULL
1830
set TQ bitvec
clear partial exec bitvec
initialize row counter
1832
start(current DFO)
1834
return
1836
start underlying source

FIG. 18B

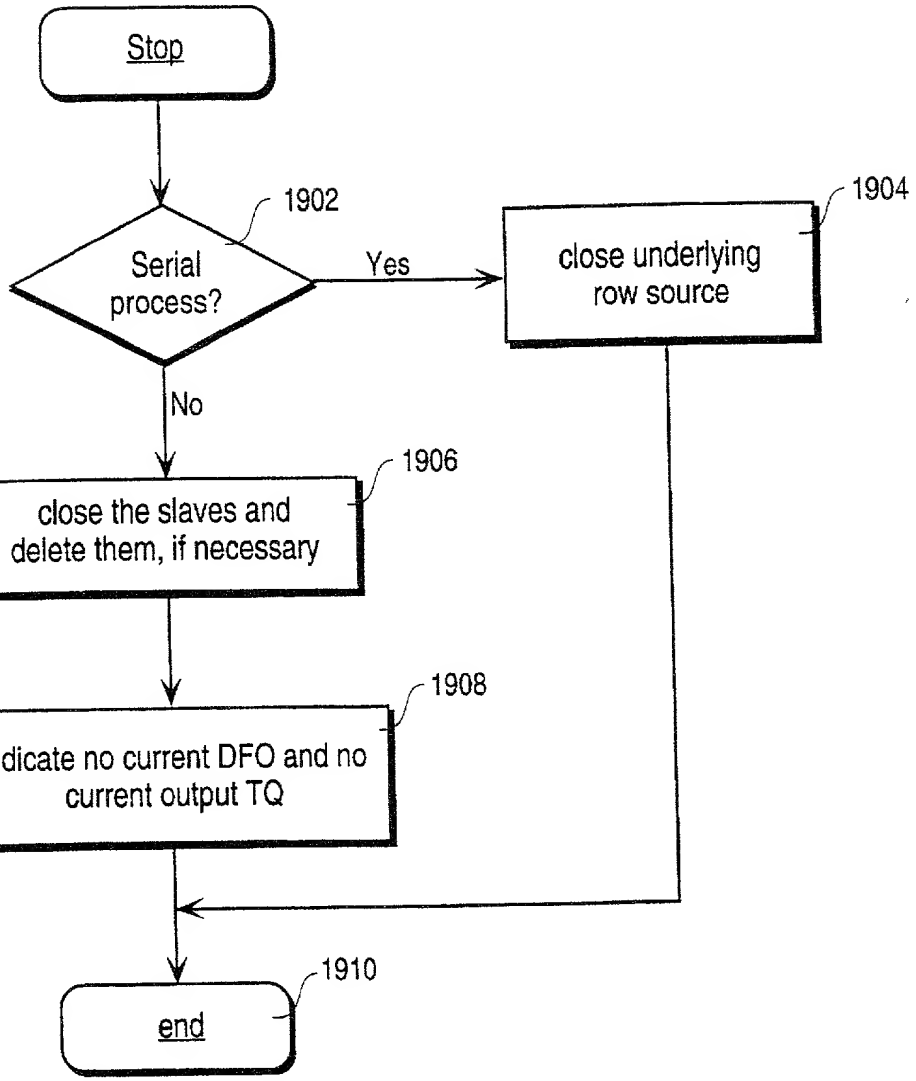
Stop
1902
Serial process?
Yes
1904
close underlying row source
No
1906
close the slaves and delete them, if necessary
1908
indicate no current DFO and no current output TQ
1910
end

FIG. 19